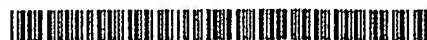


(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number
WO 01/06435 A1

- (51) International Patent Classification⁷: G06F 17/60 (71) Applicant (for all designated States except US): E-DIALOG, INC. [US/US]; 1646 Massachusetts Avenue, Lexington, MA 02420 (US).
- (21) International Application Number: PCT/US00/19403 (72) Inventor; and (75) Inventor/Applicant (for US only): ESTES, Anthony, D. [US/US]; 4 Westmoreland Avenue, Arlington, MA 02174 (US).
- (22) International Filing Date: 14 July 2000 (14.07.2000) (74) Agent: FEIGENBAUM, David, L.; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).
- (25) Filing Language: English (81) Designated States (national): AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (26) Publication Language: English
- (30) Priority Data: 09/353,896 16 July 1999 (16.07.1999) US
- (63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application: US 09/353,896 (CON) Filed on 16 July 1999 (16.07.1999)

[Continued on next page]

(54) Title: DIRECT RESPONSE E-MAIL

WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:

- + PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
- + IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
- + HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
- + CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

- A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
- B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
- C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
- D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
- E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR \$89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 1202.

(57) Abstract: An e-mail message [10] is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to complete the transaction.

WO 01/06435 A1



(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— With international search report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DIRECT RESPONSE E-MAILField of invention

5 This invention relates to direct response e-mail.

Background of the Invention

 In direct response e-mail, a vendor, for example,
can sell a product to a customer by sending an e-mail
message to the customer that describes the product and its
10 price. The customer can order the product by returning an
e-mail (sometimes called a direct response e-mail) that
gives appropriate order information. The vendor can confirm
the order by a return e-mail. The order information
returned by the customer can sometimes be determined
15 automatically using software that analyses the customer's
reply e-mail.

Summary of the Invention

 In general, in one aspect of the invention, an e-
mail message is analyzed to derive response information
20 concerning a commercial transaction. Based on the derived
information, commercial transaction data are automatically
generated in a format that is usable to automatically
complete the commercial transaction.

 In general, in another aspect of the invention, an
25 e-mail message is sent to a customer offering a product or
service for sale. The e-mail message includes locations for
response by the customer to indicate his intention to order
the product or service. The customer returns an e-mail
message that includes the response. Based on the received
30 e-mail, order information is automatically generated in a
format usable automatically by an order fulfillment system
to cause the order to be filled.

 In general, another aspect of the invention includes
automatically identifying response information which

requires resolution of an issue with the source of the e-mail message and automatically managing an e-mail dialog with the source to resolve the issue.

5 In general, in another aspect, the invention features automatically sorting e-mail messages, based on response information contained in the messages, into e-mail messages that can be processed automatically to generate commercial transactions, e-mail messages in which the response information is inadequate to permit generation of
10 commercial transactions, and e-mail messages that may be subjected to exception handling to yield information that is sufficient to generate commercial transactions.

In general, in another aspect, the invention features automatically generating a confirmatory e-mail
15 message to the source of the e-mail message confirming that a commercial transaction has been or will be completed.

In general, in another aspect, the invention features receiving inbound e-mail messages that result from corresponding outbound e-mail messages associated with a
20 marketing program, the inbound messages containing response information, each of the outbound messages being associated with a distinct piece of the marketing program. The response information in each of the inbound messages is automatically associated with the corresponding distinct
25 piece of the marketing program.

In general, in another aspect, the invention features automatically merging response information with corresponding information in a database for use in completing transactions.

30 In general, in another aspect, the invention features identifying inbound e-mail messages that cannot be processed automatically to generate commercial transactions,

and using the database information to assist in exception handling of the identified inbound messages.

Other advantages and features will become apparent from the following description and from the claims.

5 Brief Description of the Drawing

Figures 1A through 1C and 2A through 2B show e-mail messages.

Figure 3 is a block diagram of a direct response e-mail system.

10 Description of the Preferred Embodiments

Outbound e-mail messages

The two e-mail messages shown in figures 1A through 1C and 2A through 2B are examples of outbound messages associated with commercial transactions.

15 The example message 10 shown in figure 1 offers Harvard Business Review products. Message 10 includes basic copy 12 that is similar to basic direct marketing copy of the kind that is commonly used in e-mail marketing. Message 10 also contains a section 14 giving instructions on how to
20 order the products.

Inbound e-mail messages

To take advantage of the offer shown in figure 1, the recipient creates a reply e-mail message (the direct response message) and types the letters of the items that he
25 wants to order in the first line of the body of the message. In other examples, the letters could be typed in the subject line or the last line of the body of the message. The user is also asked to correct and complete shipping and e-mail address information that has been merged into the outbound
30 e-mail message in a section 16. In section 16, each of the entries is bounded by brackets. Another section could

contain merged billing information, not shown. The person who replies to the e-mail (the customer) is meant to include the corrections or additions within the indicated brackets.

5 By allowing the recipient to take advantage of the offer simply by replying to the e-mail, rather than requiring the recipient to place an order by linking to a related web-site or to print the e-mail message and FAX it back, or to call an 800 number, a much higher return rate can be achieved. For conventional outbound e-mail messages
10 that require the recipient to click on an embedded URL to go to a web site, the returns may be on the order of several hundred percent on investment (the fee charged for delivering the outbound messages). By enabling the recipient to provide direct response e-mail messages as in
15 figure 1, the return on investment can be as high as several thousand percent.

Figures 2A and 2B illustrate a similar outbound e-mail message in which there is no choice of products but only a single offer to be accepted or rejected. To take
20 advantage of the offer, the recipient types "yes" in the subject line. In figure 2B, a shipping block 18 of the kind mentioned above is shown. (In this case, the shipping block contains no information because the shipping address is the same as the billing address.)

25 One reason for including differential billing and shipping blocks is to acquire information in the return e-mail message that is similar to information captured in orders placed on a related web site. In a system in which web-site orders generate fields that can be fed directly to
30 an automated order fulfillment process, it is useful to make the e-mail message information field-wise consistent to permit the information to be delivered automatically to the same order fulfillment process.

Exception processing

Processing the inbound e-mails (the ones with responses concerning commercial transactions from the recipients of the outbound e-mails) may require custom
5 interaction with the recipients. For example, the wording of the outbound messages may be confusing to the recipients.

As shown in figure 3, the system 40 enables the transactional e-mail message processor 42 to determine when a dialog with the recipient 44 is needed and then assists a
10 human service representative 46 to conduct an effective dialog 48. The dialog can be conducted on behalf of the vendor 50 but without involving the vendor. Alternatively, the vendor's fulfillment process 52 can be notified electronically 54 of interaction that may be required.
15 Easing the processing of responses that include customer orders is important because the orders typically come back quickly, e.g., within 36-48 hours, and in large volume. The ability to deal with questions that arise as a result of the contact from a customer service point of view keeps the
20 vendor's customer service organization from being overwhelmed by the responses that come back.

The ability to process exceptions without involving the customer service organization of the vendor is based partly on knowing how the outbound e-mail messages were
25 constructed. As a simple example, a recipient may ask an unnecessary question that could have been answered by reading the outbound e-mail message. The e-mail message processor can pull out the relevant portion of the message and send it back to the recipient to answer the question.

ProcOrder Process

The inbound e-mail messages 60 are batch processed by a script called ProcOrder 62. ProcOrder parses the elements of the inbound e-mail messages in accordance with the original set up and instructions of the outbound e-mail messages 64. ProcOrder determines if all of the items that are required for an order to be completely processed automatically appear in the inbound e-mail message. For example, the script would look for the ordering token, such as the word "yes" or a series of letters depending on whether it is a single or multiple offer. The script would also look for footer information in the e-mail message, including a code that identifies the given campaign and the given offer, as seen in block 66 of figure 2B. In that example, there are four components in the footer, but only two are represented because the other two are not required in this instance. The first element is a customer identifier 68, e.g. 861270. Then there is a space 70 between two pipes that would contain the list identifier if there were one. There may be multiple recipient lists for a given marketing campaign. In the example, there is only one list, and there is no list identifier. A list number 243 might refer to a list of people who made a purchase at the vendor's web site or who subscribed at the web-site for a listserv.

The third footer item could be a source of awareness code 72, e.g., 3275, which identifies a particular marketing campaign. For example, in the case of figure 2, the code could refer to a Benchmarking Three-part Video Series offer.

The last item in the footer, located between the final pipe and the first right bracket would be a flight identification code 74. A given campaign could have multiple flights of e-mail messages.

After looking for the footer information, the ProcOrder parser looks for fields in the billing and shipping address blocks that are required to complete the order. What is required may vary with the type of campaign
5 but typically the minimum requirements are a name and a physical address. If the information is not completely available in the response e-mail message, the script checks to see if it is available in the database 76. If not
10 available in either place, the script generates an exception entry for an exception list. The exception list is provided to a service representative 46 who can then act on it (without involving the vendor's customer service organization), e.g., by sending back an e-mail message asking for the shipping address.

15 If all required information is available, the script generates a fully fielded valid order in a format required by the fulfillment system of the vendor and adds it to a batch of valid orders 78 which are sent electronically to the fulfillment process.

20 Confirmation e-mail message

As a result of running the ProcOrder script, an e-mail message 80 is returned to each customer either to confirm an order or to request more information. In the latter case, a dialog ensues and is managed by software and
25 through an exception handling service as explained earlier. For example, the customer's response could say something like "sure, send"; or "send it and I'll take a look." Shortly thereafter the customer would receive a confirmation
"Thank you for your order; you can expect the CD-ROM in
30 about seven business days. Please let us know if there is anything else we can do to help simply by replying to this e-mail."

One-click ordering

Another feature of the e-mail dialog with a customer involves simplifying and optimizing the presentation of content. In the examples of figures 1 and 2, the information is presented in a simple text format. It is useful also to provide in-line HTML code in the outbound e-mail message in a manner similar to the one-click ordering that Amazon.com offers in a web-site context. In one-click ordering, the customer sets up an account by providing credit card and shipping information. On subsequent visits to the web site, the customer can pick a product with one click, place an order, and have it shipped. A similar technique could be adapted to e-mail message interchange by embedding one-click ordering into e-mail.

An advantage of in-line HTML code is the opportunity for a much higher response rate because of the higher graphical contact and higher level of engagement normally achieved by a graphical message.

Template

The outbound e-mail messages are set up in a standard format using templates 90. The templates enable either a single-offer message or a multiple-offer message. Other templates are also possible, including one that embeds in-line HTML into the message as mentioned above, either for the single-offer or multiple-offer cases.

In addition, a set-up tool 92 permits the parameters of a given campaign to be defined, including the source of awareness code, the flight identification code, the campaign identification code, and similar information. The set-up tool also permits defining the tokens that are to be used in a given campaign (for example, the letters assigned to different products being offered). The set-up tool also allows a definition of the required fields that must appear

in a given campaign to enable automated generation of orders to an existing fulfillment system.

The set-up tool also provides a user interface that enables a vendor to help in entering the set-up information.

5 The result of applying the tool to the templates is a set of outbound message forms 94 that are ready for use.

Reporting tool

After the template is set up and the system is ready to launch a flight, address 108 and other information 110.

10 112 stored in the target list of customers is merged with the message forms, and the e-mail messages are automatically generated and sent by an outbound e-mail delivery engine 96. Customers then begin to respond. The ProcOrder script generates automatic orders to the fulfillment system and
15 exception information for additional processing.

A reporting tool 104 aggregates information about the responses for a given campaign according to source of awareness code and flight. The information is made available on-line to the vendor and can be used for a
20 variety of marketing purposes. The information could be generated as an Excel file attached to an e-mail, or as a paper-based report, or as an electronic file that is transferred on a batch basis.

Gathering additional information from database

25 There may be an intermediate step between the parsing engine's (ProcOrder) extraction of information from an e-mail message and the generation of the valid order. The intermediate step could be a querying process 112 to gather additional information from an existing database.
30 The additional information may not have been included in the outbound e-mail messages but may be needed to generate a valid order. For example, product codes 112 may be stored in the database but not included in the outbound e-mail

message. The letters entered by the customer can be mapped to the actual product codes by reference to the tables of the database based upon the source of awareness code.

5 The resulting valid order is a fully-fielded record that has the fields required by the client's order fulfillment system to process an order.

Exception treatment

10 Exception handling can be treated in different ways depending on the circumstances. For example, an exception might occur when a customer responds from an e-mail client that does not quote the original text of the outbound e-mail message. The inbound e-mail message then has the customer's e-mail address, a subject line that says "yes", and the original subject line from the campaign, but does not have
15 the required information for the shipping address or the footer information. ProcOrder would kick that out as an exception, but the exception handling system would allow a response management representative 46, based on the e-mail address, to confirm, from the database 76, that all of the
20 required information is available. Use of the subject line allows the system to tie back to the appropriate campaign and to figure out who is ordering and what he is ordering. A valid order can be created without further interaction with the customer other than to send him a confirmation that
25 the system has been able to enter a valid order on his behalf.

The system thus recognizes that it is not likely to be possible to automate every interaction with the customer, but it may be possible to complete a dialog with essentially
30 all of the customers from whom inbound e-mail messages are received by automatically identifying messages that will require custom human handling and providing information and

tools that enable the human handlers to complete the exception transactions in an efficient manner.

Non-order response processing

Not every inbound e-mail message is an order. Non-order messages include undeliverable bounced messages to ad hoc customer service responses. Non-order inbound e-mail messages must be identified by the parsing engine.

Undeliverable e-mail messages 114 are automatically separated from the inbound e-mail stream and stored for offline handling by a human response handling professional, who operates a script on the files of undeliverable messages. The script classifies them as "soft" and "hard," parses e-mail addresses and footer data from the messages, matches the parsed records to the database, and flags appropriate records as "undeliverable".

Other non-order messages also are handled manually as explained earlier.

Vendor creation of e-mail campaigns.

A campaign creation tool 126 is provided to a vendor to enable simple entry of all information needed to create an e-mail campaign, including all the parameters, the text of the messages, and the tables of data needed in the database. The vendor delivers the campaign electronically to the transactional e-mail processor which then delivers the e-mail messages, receive the responses, processes all exceptions, and returns to the fulfillment system the vendor orders in a proper format.

A web-based vendor interface 128 enables on-line viewing by the vendor of the status of all campaigns, including the state of those that are in development and the results of those that are "live". The information is hosted by the transactional e-mail processor in part based on the

database 76. The interface also gives the vendor a mechanism to check text and other content into the database.

Alternatively, instead of automatically permitting the vendor to fully create a finished campaign, the vendor
5 may be enabled to download and check into the database a proposed campaign. Then an account executive of the e-mail handler process would review it and work with the vendor to complete it before it is finally queued for distribution.

Appendices A, B, and C contain more detailed
10 descriptions of aspects of implementations of the invention. Appendix D contains source code written of an example of the ProcOrder process.

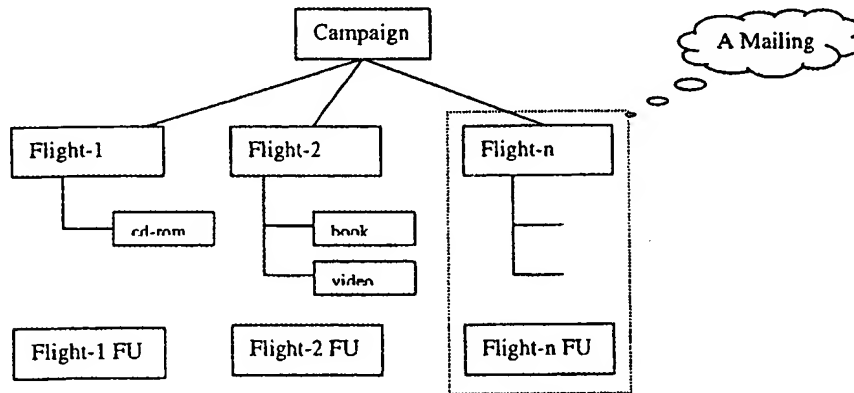
Other implementations are within the scope of the following claims.

15 What is claimed is:

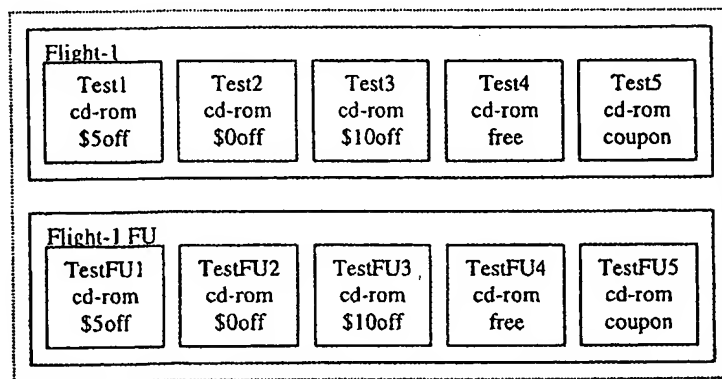
e-mail Protocols, Structures, Definitions & Cycles

MAIL CAMPAIGN

A mail campaign is defined in terms of **Flights**.



A flight contains one or more items that are being promoted but is not restricted to the manner in which the promotion is performed. A flight could be setup to promote a CD-ROM but has variances in the promotion of the item to the targeted mailing. These variances in a flight are called **Tests**.



In order to reference the entities above, each are assigned Ids as follows:

APPENDIX A

A Campaign

- Has a Campaign ID , **CID**
- Has 1 to n flights and flight follow-ups

A Mailing

- Has a Mailing ID, **MID**
- Has a Client Code (SOAC). The mailing is performed on behalf of the client.
- Has a Flight ID, **FID** and a Flight Follow-up ID, **FID**

A Flight

- Has a flight ID, **FID**
- Shares a **SOAC** with its Flight Follow-up
- Contains one or more Test Ids, **TID**
- Has 1 to m items for sale

An Item

- Has an Item ID **IID**
- Has an Item code (determined by the client)

To summarize, the following ID acronyms are used:

- | | | |
|--------------|---|---|
| - MID | - | Member ID (ID of the person receiving the e-mail) |
| - CID | - | Campaign ID |
| - LID | - | maiLing ID |
| - FID | - | Flight ID |
| - TID | - | Test ID |
| - IID | - | Item ID |
| - SID | - | Style ID (ID of the style of the e-mail message) |

MAIL PROTOCOLS

SMTP

The *Simple Mail Transfer Protocol* (SMTP) is described in RFC 821 and is the way that two sites on the Internet exchange mail messages.

The commands are:

- HELO *domain*
- MAIL FROM: *username*
- RCPT TO: *username*
- DATA
- QUIT

Each command is terminated with a CR-LF pair. Replies start with a three-digit response code and continue with text designed to be read by users.

POP3

POP3 is the Post Office Protocol. If the site is always on the Internet, then mail would be sent with an SMTP-sender and received with an SMTP-receiver. However, in many cases it is not possible to maintain a permanent internet connection and in such cases, the Post Office Protocol is used to receive the inbound mail. POP3 allows mail to be stored on machine that is always on the Internet and a receiving host connects to it, asks for any mail and disconnects.

The commands are:

- USER *username*
- PASS *password*
- STAT
- LIST *[message number]*
- RETR *message number*
- DEL *message number*
- LAST
- QUIT

THE ANATOMY OF AN OUTBOUND E-MAIL MESSAGE

An outbound email message has a predefined structure. The message is created by combining together a text block with an email address.

The structure of an outbound mail message consists of a Property, Address, Subject, Header, Body, Personal and Token.

Headers
Address
Subject
Salutation
Body
Personal
Token

Structure of an outbound e-mail message

a) Headers

The first few lines of an Outbound Mail message are called the Headers and have a defined format. This information is not normally displayed to the User by a Mail Client application and can only be viewed if the Client permits e.g., in Microsoft Exchange this information can be viewed by listing the properties of a mail message as follows:

```
Received: by mail.kersur.net (mbox peterk)
(with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 20:59:00 1999)
X-From: aestes@e-dialog.com Fri Mar 19 13:43:54 1999
Return-Path: <aestes@e-dialog.com>
Received: from montana.e-dialog.com (mail.e-dialog.com [207.31.244.21])
by mail.kersur.net (8.9.1/8.9.1) with ESMTD id NAA10659
for <peterk@sytech.com>; Fri, 19 Mar 1999 13:43:53 -0500 (EST)
Received: by MONTANA with Internet Mail Service (5.5.2448.0)
id <GZZPKKZ2>; Fri, 19 Mar 1999 13:43:57 -0500
Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47F10B@MONTANA>
From: Anthony Estes <aestes@e-dialog.com>
To: "peterk@sytech.com" <peterk@sytech.com>
Subject: FW: Warning: could not send message for past 4 hours
Date: Fri, 19 Mar 1999 13:43:54 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: multipart/mixed;
boundary="-----_NextPart_000_01BE7238.71C0EB9A"
```

This message is in MIME format. Since your mail reader does not understand this format, some or all of this message may not be legible.

```
-----_NextPart_000_01BE7238.71C0EB9A
Content-Type: text/plain;
charset="iso-8859-1"

-----_NextPart_000_01BE7238.71C0EB9A
Content-Type: application/octet-stream;
name="ATT00547.TXT"
Content-Disposition: attachment;
filename="ATT00547.TXT"
```

-----=_NextPart_000_01BE7238.71C0EB9A
 Content-Type: message/rfc822

Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47EFF2@MONTANA>
 From: Anthony Estes <aestes@e-dialog.com>
 To: "Hillary Gaeth (E-mail)" <HGaeth@engage.com>
 Subject: * Thurs, Fri ?
 Date: Tue, 9 Mar 1999 08:04:20 -0500
 MIME-Version: 1.0
 X-Mailer: Internet Mail Service (5.5.2448.0)
 Content-Type: text/plain

-----=_NextPart_000_01BE7238.71C0EB9A--

A header starts with a *field name* followed by a colon and the field body. The contents of the field body may be rigidly defined or free form.

The following headers are mandatory; that is, there must be a header with each of these names:

- Date
- From, or Sender and From
- To, or CC (carbon copy), BCC (blind carbon copy)

The following headers are optional:

- Return-path
- Received
- Reply-To
- Message-ID
- In-Reply-To
- References
- Keywords
- Subject
- Comments
- Encrypted

Some of these headers are obscure and rarely used. In addition, some mail clients generate their own extra headers. Many such extra header start with the characters "X-" because if extra mail headers are added to the RFC they will never start with these characters.

A header may be split over two lines according to the following rules:

- The split must be at a place where whitespace(blank or tabs) would normally occur; for example, not in the middle of a username or similar field.
- The continuation line must start with a space or tab.

Similarly, a header can have extra white spaces almost everywhere except in the middle of a field.

Two headers that are given special attention in the context of this document are those that contain the address and subject.

b) Address

The address is a header that contains the email address of the person receiving the message.

c) Subject

This is a header that describes the subject of the mail. Contained in the subject is a **Subject Token** in the form of two characters. For example, for the mail subject '** A Special Offer for Selected Managers', the Subject Token is **

The Subject Token is chosen to identify the Test ID of the mail message.

d) Salutation

The salutation is a text block in the outbound mail message that is personalized to the person receiving the message. The intention is to provide a personalized greeting and an indication of the sender.

From: Harvard Business School Publishing <hbasp@e-care.com>
To: 'esalas@protelsa.com.pe'
Subject: ** A Special Offer for Selected Manager
Date: Tuesday, July 14, 1998 6:47 PM

From the Desk of Laura Winig
Harvard Business School Publishing Corporation
Boston, Massachusetts

Tuesday, July 14, 1998

Ms. Elizabeth Salas
5700 Collins Avenue Apt 6h
Miami Beach, FL 33140-2308

Dear Ms. Salas:

The information in italic is personalized and is combined with the remainder of the message at the time of dispatch.

e) Body

The body contains the main core of the message. The format and layout is fixed and not personalized for a particular test.

Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-Obligation Test Drive!

Do you work with colleagues and clients in multiple locations? Communicate more by email and conference calls than through meetings? Find your "office" is wherever you are at any given moment? Then you're working "virtually" and you know that working effectively without proximity is essential in today's workplace. And you know it's not as easy as it looks.

Here at the publishing arm of Harvard Business School, we've combined extensive research and real-life examples to create "Virtual Work: Real Results"--a dynamic multimedia program that can improve your effectiveness working "virtually."

Use this engaging tool and you'll understand:

- * the dynamics and politics of working virtually-
and how to handle tough situations when you can't be face-to-face;
- * how to most effectively use email, video conferencing,
voice conferencing-and how to overcome fear of technology;
- * ways to build relationships and trust without "human touch"-

and how a virtual team can work efficiently and seamlessly.

 We bring you tricks of the trade from the experts in working virtually.
 Then, you gain confidence using these techniques through an interactive case study where you lead
 a virtual team through a project. You'll make decisions that determine the success of its
 efforts-all in a realistic, but no-risk, environment.

Overcome the isolation and conflicting loyalties that are inherent in working in a virtual
 environment-and get ready for success with "Virtual Work: Real Results."

 Take "Virtual Work: Real Results" for a No-Obligation Test Drive.

 Simply reply to this email and we'll send you the program with our compliments. We're confident
 you'll find you're working more effectively in the virtual world. After 14 days, we'll send you
 an invoice for just \$295 (single user license).

But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
 to pick it up. You will owe nothing.

Sincerely,

Laura Winig
 Director

In the above example the response with a "YES" is sufficient to indicate the purchase of the
 single product on offer. In a multi-product offer, the items would be listed and associated with a
 letter. For this type of mailing, the responder would list the letters in the response.

A mailing may have a follow up 'reminder' flight. This reminder would not go to respondents of
 the original flight. For example:

On Wednesday, July 1, I sent you a special offer on "Virtual Work: Real Results": a new
 interactive CD-ROM from Harvard Business School Publishing.

Since I haven't heard back from you, I wanted to send you a reminder Before the offer expires.

If you are simply not interested, I apologize for the intrusion.

----- BELOW IS A REPRINT OF THIS SPECIAL OFFER -----
 Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
 Obligation Test Drive!

 Do you work with colleagues and clients in multiple locations? Communicate more by email and
 conference calls than through meetings? Find your "office" is wherever you are at any given
 moment? Then you're working "virtually" and you know that working

...
 ...

confident you'll find you're working more effectively in the virtual world. After 14 days, we'll
 send you an invoice for just \$295 (single user license).

But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
 to pick it up. You will owe nothing.

Sincerely,

Laura Winig
 Director

f) Personal

The personal is added to the outbound message, after the mail body. The purpose of the text
 block is to request personal details from the respondent. The information requested is presented

in two columns, the first indicating the type of information required and the second as a place for the reply to be entered (in [..]). For example,

```

FIRST NAME:      []
LAST NAME:       []
TITLE:           []
COMPANY:         []
DEPARTMENT:      []
ADDRESS1:        []
ADDRESS2:        []
ADDRESS3:        []
CITY:            []
PROVINCE/STATE:  []
POSTAL/ZIP CODE  []
COUNTRY:         []
PHONE:           []
FAX:             []
EMAIL:           []

```

These details are to determine the shipping and billing information.

e) Token

The token follows the Mail Body (or Mail Personal if applicable) and contains information about the mailing and also the addressee e.g., [[878119|2815|1]]

The format of a Mail Token is

[[MID | TID | SID]]

where

- MID is an membership ID assigned to the person receiving the outbound message
- TID is the test ID assigned to the outbound message. This would be related to the Subject Token.
- SID is the style ID assigned to the style of the mailing i.e., single product or multi-product.

These three pieces of information uniquely identify the receiver of the outbound message and also the information they received. Hence, the processing of a response is greatly simplified if the reply returns the mail token.

In summary, an outbound message contains generic information that is the same in all the mailings of a test, and also personalized:

Generic
Subject
Body

Personalized
Header
Address
Salutation
Personal
Token

THE ANATOMY OF AN INBOUND E-MAIL MESSAGE

An inbound email message has a predefined structure. However, the structure may not be 'structured' sufficiently for it to be automatically processed.

Header
Address
Subject
Response

a) Headers

The first few lines of an Inbound Mail message are Headers and have a defined format. This information is similar in format to that of the outbound message. If the response is produced by replying to an outbound message, (instead of creating it from scratch) then it is probable that the headers of the outbound will be in the inbound mail.

b) Address

The address is a header that contains the email address of the respondent.

c) Subject

This is a header that describes the subject of the mail. This is free-form text and no assumptions can be on its structure. It may be the subject that was used in the outbound mail.

d) Response

The response is a text block containing the message from the respondent. It should not be assumed that the response has any structure since a responder has the freedom to write a reply in "free format" and is not forced to a guideline. This creates a number of problems for the processing of an inbound response since rigid rules cannot be applied.

The points that can be noted about a response are that:

- All responses will contain headers.
- All responses will contain the responders email address.
- All responses will contain a subject. The textual content of the subject cannot be assumed since it can be freely edited and so may not resemble the content of the outbound. If a subject contains a Subject Token then it should correspond to the TID.
- If the response contains the Mail Token then the MID, TID and SID will be available and consequently it will be clear on the approach that should be taken in processing the response i.e., the handling of a single product 'YES' versus a multi-product 'ADG'.

AN E-MAIL LIFE CYCLE

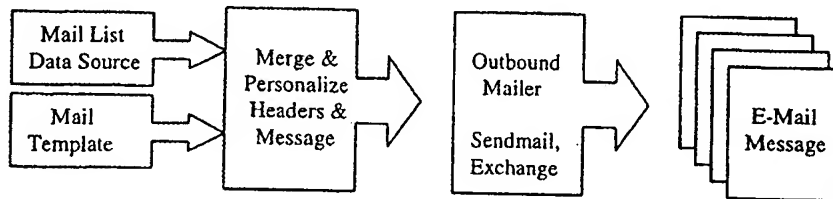
OUTBOUND

a) Create an outbound message

An outbound email message consists of texts block that contains information and one or many questions or choices. The text block of an outbound mail message is sometimes referred to as a **Mail Template**.

b) Deploy the outbound message

The message can be deployed to anyone for whom an e-mail address is available. The address information in a **Mail List** (e.g., database, listserve) is merged with the **Mail Template** and sent. In some cases, the X-Headers are also personalized to reflect the purpose of the outbound mail message.



INBOUND

a) Receive a Reply

The content of a reply can have various forms

- The reply responds to the outbound message e.g., an order.
- The reply requests to unsubscribe from future mailings. The reply contains an indication that the respondent does not wish to receive any further mailings e.g., UNSUB, UNSUBSCRIBE, UNJOIN and REMOVE.
- The reply requests customer service and ad hoc questions.
- The reply indicates a change in personal details or to be added to the mailing list.

There is also the possibility that the outbound message did not reach its final destination and that it bounced back. There are two categories of bounces, hard bounces and soft bounces.

- A hard bounce notification indicates outright failure. For a hard bounce, the subject would contain a message of the following form:

Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)

The sender of a hard bounce is usually specific such as:
Mail Delivery System (MAILER-DAEMON)

A hard bounce can also be detected in the response X-Header:

```
Received: by mail.kersur.net (mbox peterk)
(with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 21:04:39 1999)
X-From_: MAILER-DAEMON Fri Mar 19 21:04:36 1999
Return-Path: <MAILER-DAEMON>
Received: from localhost (localhost)
  by mail.kersur.net (8.9.1/8.9.1) with internal id VAA20320;
  Fri, 19 Mar 1999 21:04:36 -0500 (EST)
Date: Fri, 19 Mar 1999 21:04:36 -0500 (EST)
From: Mail Delivery Subsystem <MAILER-DAEMON>
Message-Id: <199903210204.VAA20320@mail.kersur.net>
To: <peterk@sytech.com>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
  boundary="VAA20320.921981876/mail.kersur.net"
Subject: Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)
Auto-Submitted: auto-generated (failure)
```

The response body can also contain failure information:

The original message was received at Fri, 19 Mar 1999 21:04:35 -0500 (EST)
from dialup111.kersur.net [207.180.95.76]

----- The following addresses had permanent fatal errors -----
<JohnSmith@SSOOFFTTEECCHH.com>

----- Transcript of session follows -----
550 <JohnSmith@SSOOFFTTEECCHH.com>... Host unknown (Name server: ssoofftteecchh.com: host not found)

- Softbounces can be of three types:

- *NonDeliveryNotification* occurs when a given message has not been delivered yet but will continue to try and deliver for a further specified period of time. This state can be detected in the response subject:
FW: Warning: could not send message for past 4 hours

The response can also be detected in the response body:

```
*****
**      THIS IS A WARNING MESSAGE ONLY      **
**  YOU DO NOT NEED TO RESEND YOUR MESSAGE  **
*****
The original message was received at Tue, 9 Mar 1999 08:00:18 -0500 (EST)
from mail.e-dialog.com [207.31.244.2]
```

----- The following addresses had transient non-fatal errors -----
hgaeth@andexc01.cmgi.com
(expanded from: <HGaeth@engage.com>)

----- Transcript of session follows -----
hgaeth@andexc01.cmgi.com... Deferred: Connection refused by
andexc01.cmgi.com.
Warning: message still undelivered after 4 hours
Will keep trying until message is 3 days old

- *AutoResponders* are notifications which actually indicate delivery but are sent by mail agents to indicate that the user will not be able to respond immediately (possibly on vacation) but the sender should expect a response when they return.
- *Unknown*

b) Process the reply

Automatic processing of an e-mail response is defined as the ability to determine accurately the requirements of the reply by using text inspection and search rules.

To automatically process an e-mail response, there are a number of criteria that need to be satisfied by the content of the reply. The main three criteria are:

- ci) Who is the Respondent
- cii) What outbound message is the response to
- ciii) What does the Respondent require

The exception to the above are hard and soft bounces that are identified by other e-mail properties.

Since all legitimate mail responses will contain the e-mail address of the Sender this can be regarded as a base information for all responses. This information fulfills criteria ci) but alone is not sufficient for a response to be processed automatically.

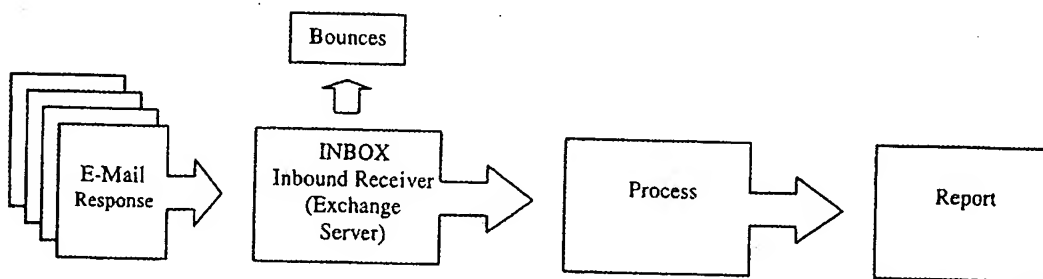
Criteria cii) can be satisfied by the Subject Token, provided the respondent has not altered it. Note that the Style ID is implied if criteria cii) is satisfied.

Another source of information is the Mail Token. For a reply that contains a Mail Token, both criteria ci) and cii) can be determined.

But how is criteria ciii) satisfied? There is no simple solution to this question since the answer lies in the body of the reply, which is in 'free form'. The only method available that can determine the requirements of the respondent is to parse the email reply based on the SID.

c) Produce Reports

After determining the requirement of the inbound e-mail, database entries and reports can be produced.



Design Specification for QuickReply

QUICKREPLY

The QuickReply application is used to process e-mail responses to mailing campaigns. The building and sending of the e-mail messages is not within the scope of QuickReply.

The test within a campaign's flight provides the outbound e-mail message. The message will contain both generic and personalized information and will normally require a response from the person the message is sent to.

Responses to a test are received in the Inbox folder of the mailbox. The design objective of QuickReply is to read the responses from the Inbox, and decide from its content the requirements of the response. On determining (or not determining) the requirement, the response is filed to pre-configured sub-folders and the appropriate tracking database is up-dated.

QuickReply will generate diagnostic information that will support the reasoning on how it reached its decision.

The main data storage area of QuickReply is Microsoft Access relational database. Appropriate records will be added to this database as the response processing is performed. In addition, it is likely there will be responses that cannot be automatically processed and, in such cases, Customer Service will use QuickReply in manual mode. Manual mode will also add records to the appropriate tables in the database.

The information in the database will be used to generate reports.

APPENDIX B

MAILBOX ORGANIZATION

Mailings are performed on behalf of a Client. Consequently, all e-mail transfers for a particular Client will be performed within the Clients mailbox.

The organization of the Clients mailbox includes an Inbox for the responses and a pre-configured number of sub-folders where the responses are filed.

Mailbox - <Client Reference e.g., HBSP>

Campaign CID

Flight FID

AC

YYMMDDhh

AddChange

YYMMDDhh

BouncesHard

YYMMDDhh

BouncesSoft

YYMMDDhh

Master

Order

YYMMDDhh

Unclear

YYMMDDhh

Unsubscribe

YYMMDDhh

Inbox

YYMMDDhh

In the above, YYMMDDhh is a directory named after the year, month, day and hour. This format reflects the date and time the entries were placed in that folder.

Each test flight will have a number of responses associated with it. When QuickReply categorizes a response, it will be automatically moved from the Inbox to the appropriate sub-folder.

- **AC** Responses for customer service to process manually
- **AddChange** Responses that contain personal detail changes
- **BouncesHard** Hard bounces
- **BouncesSoft** Soft bounces
- **Master** Master version of the outbound mail
- **Order** Responses with an order
- **Unclear** Responses that are unclear as to how they should be processed
- **Unsubscribe** Response to unsubscribe

E-MAIL STYLES

The Style ID (SID) of a response indicates the approach that should be taken by QuickReply during processing.

The SID is a two-digit numerical value

XY

where:

- X=1 indicates the outbound Personal block is included in the outbound message.
- Y=message layout style described below.

The following two layout styles are supported:

a) Single Product Offer – layout style=1

This style offers one product. The respondent is asked to reply "YES" if they are interested.

b) Multi-Product Offer – layout style=2

The mailing information is based on information in a relational database. This style offers a number of products the respondent can choose from. For example, the outbound message could contain:

Your choices (detailed below) are:

- A: The work of Leadership Audio
- B: Leading Your People Audio
- C: Leading Change Successfully Audio
- D: Overcoming Resistance to Change Audio
- E: Gaining Competitive Advantage Audio

The respondent makes a choice and responds by stating their requirement in the free format e.g., "please send me ABE".

The number of items can vary as well as the choice symbol i.e., 1,2,3, could be used instead of A,B,C.

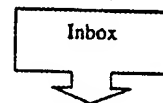
PROCESSING INBOUND MAIL AND REPORTING

The general processing and reporting function is described below. The general flow in processing responses is to determine the membership ID (MID), then the test ID (TID) and style ID (SID) and then perform an examination of the response to determine the requirement.

The processing criteria can be summarized as follows:

- ci) who is the Respondent?
- cii) what is the response to?
- ciii) what does the Respondent require?

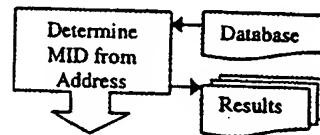
All inbound responses are targeted to the Inbox of the Client's Mailbox. On a periodic basis, the responses in the Inbox will be processed.



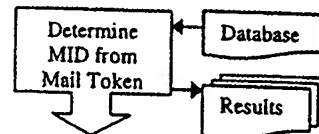
ci) Who is the respondent? – Determine the MID

There are several ways in which the Membership Information can be determined. The importance of the MID is that it could provide a handle to the Member's personal information.

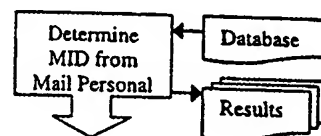
a) Extract the Response Address from the response. Determine if a unique record exists for the Address. If a unique address exists then the MID is determined. If the record is not unique then examine the mail token.



b) If the Mail Token exists then the MID is determined. Use the MID to confirm that the respondent's e-mail address is the same as that in the database. If the addresses are different then report that this was the case but continue to process.



c) If the SID indicates that the Personal block was included in the outbound, then extract the Member details from the Mail Personal. The extraction assumes that the response information is in square brackets '['']. The field definitions are defined in the database together with an indication of those that have to be completed by the responder. If Mail Personal is not complete (the fields are indicated in the database), move the response to Unclear and report.

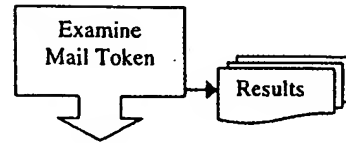


d) If the MID cannot be determined, move the response to Unclear and report.

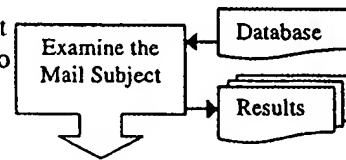
cii) What is the response to? – Determine the TID and SID

The next stage in response processing is to determine what the response is to. This can be determined from the Test ID.

a) If the Mail Token exists then extract the TID and SID. If the Mail token is absent, report but continue the determination of the TID.



b) Determine the TID from the database based on the Mail Subject of the response. If the look-up is successful, then the SID will also be determined.

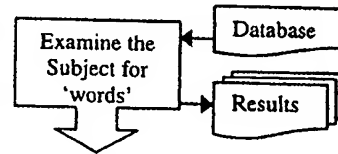


c) At this stage the TID and SID should be known. If not, move the response to Unclear and report.

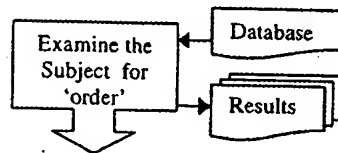
ciii) What does the Respondent require?

With the MID, TID and SID determined, the response message(s) are still of no value until it is established what the respondent requires. The obvious approach to this problem is to compare the response with the original and try to make some sense of the differences.

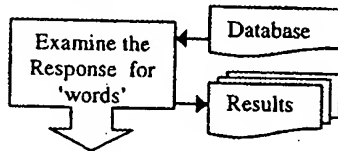
a) If the Mail Subject did not contain the Subject Token, examine for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.



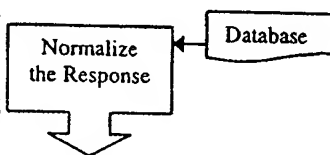
b) If the SID indicates a single item promotion, examine the Mail Subject for words that appear in the word action table. Any order responses are moved to the Orders sub-folders.



c) Examine the response for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.

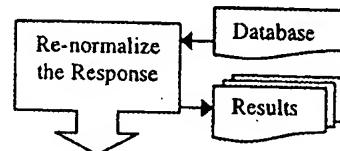


d) Remove all the text of the outbound message from the response.

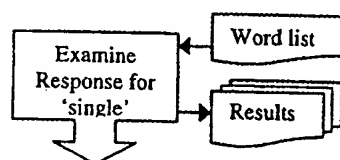


e) Normalize the response. Remove characters from the beginning of each line. The list of characters to be removed are in the word replacement table with ModeID=LineStart.

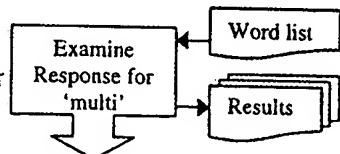
f) Replace phases from the normalized response. The phases that are replaced are given in the word replacement table with ModeID=PhaseReplace.



g) If the SID indicates that there is only a single item on promotion, examine the normalized response for words that indicate an order of a single product item. The words are in the word action table with the ActionID=SingleOrder. If order



h) If the SID indicates that there are multi-items on promotion, examine the normalized response for phrases that indicate an order of multi-items. The words are in the word action table with the ActionID=MultiOrder. In the case of multi-products, the



information the StartOption and EndOption from the bindings table are used to verify that item(s) ordered are in the correct choice range.

i) Any order requests are moved to the Orders sub-folder.

j) If the purpose of the response cannot be determined, then move the response to the Unclear sub-folder.

DATABASE ORGANIZATION

QuickReply uses two types of databases that are stored in Microsoft Access. The first database is part of the QuickReply application and contains high-level information on the campaigns, flights and tests. The second database is specific to a particular Client Test.

a) QuickReply Database

The QuickReply database contains the information that binds campaigns, flights and tests.

tblTestBindings	- contain Test bindings
Test ID	unique
Client ID	
Client Test Database	database name used for the test information
Subject	subject text of the outbound test
Style ID	the style of the outbound message
StartOption	character of the first item in the test
EndOption	character of the last item in the test
Campaign ID	
Mailing ID	
Flight ID	
tblOutboundPersonal	- contains personal details that appear in a test
Test ID	unique
Question	prompt displayed in personal block e.g., First Name
AnswerLen	the maximum length of the answer
AnswerReq?	flag to indicate that an answer must be given
tblWordActions	- contains words that convey an action
Word	unique
Action ID	action id for this word e.g., 'bounce'
tblWordReplacement	- contains phrases and their replacement
Phase	unique
Replacement	replacement to the phrase
Mode ID	type of replacement

b) Client Test Database

tblMembers	- contains member details
Member ID	unique
Email	
Prefix	
First	
Middle	

WO 01/06435

PCT/US00/19403

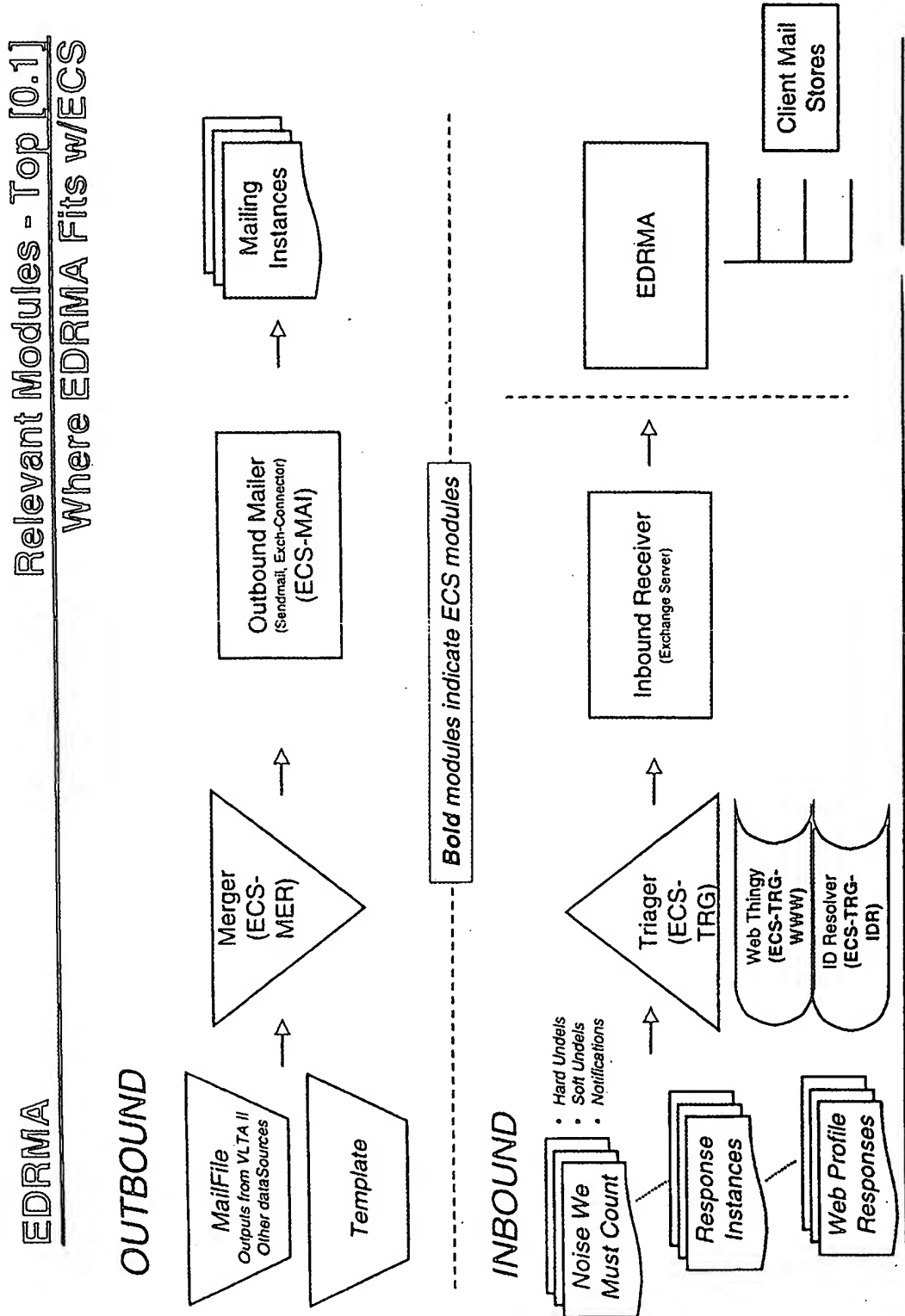
Last
Suffix
Title
Company
BillAddress1 ShipAddress1
BillAddress2 ShipAddress2
BillAddress3 ShipAddress3
BillCity ShipCity
BillState ShipState
BillZip ShipZip
BillCountry ShipCountry
Tel
Ext
Fax

EDRMA**Introduction**

This document is intended to provide a schematic synopsis of E-Dialog Response Management Architecture (EDRMA).

Thereby it is hoped that the design of the Verbind E-Mail Channel Server (ECS) can best accommodate EDRMA's input requirements, and so EDRMA's output requirements may be adjusted to match those of Verbind LifeTime's architecture (VLTA)

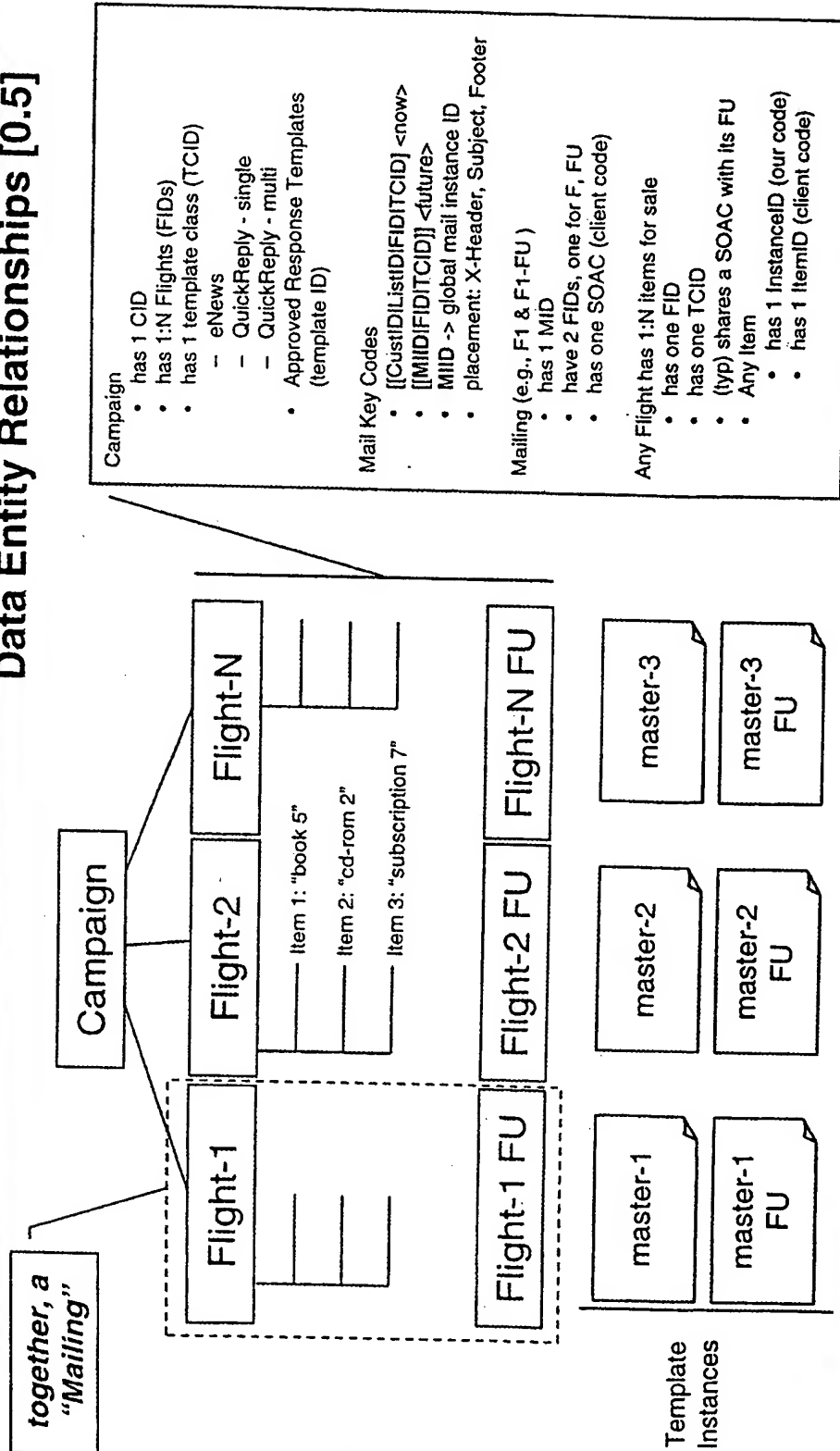
"First, some acronyms..." - *Anonymous*



EDRMA

Model

Data Entity Relationships [0.5]



EDRMA Mailbox

Canonical Data Structures - Exchange Server Data Store [0.2]

Mailbox - <ClientFullName> (not alias: "hbsp")

<CampaignName>[<CID>]

- ApprovedResponseTemplates <templateID>

- <FlightName>[<FID>]

- AC (i.e., "Advocacy Care")

- + <date: YYMMDDhhmm>

- ADDCHANGE

- + <date: YYMMDDhhmm>

- HARDBOUNCES (i.e., hard undelivered... "smith@foo.com is not a valid addressee")

- + <date: YYMMDDhhmm>

- INBOX (monolithic, this Mailbox)

- MASTER (contains master template, this FID)

- ORDERS

- + <date: YYMMDDhhmm>

- » [[custID|IsID|FlightID|CID]] << footer tags (x-header, subject)

- SOFTBOUNCES

- + DELIVERYNOTIFICATIONS

- » <date: YYMMDDhhmm>

- + AUTORESPONDERS

- » <date: YYMMDDhhmm>

- + UNKNOWN

- » <date: YYMMDDhhmm>

- UNCLEAR

- + <date: YYMMDDhhmm>

- UNSUBS

- + <date: YYMMDDhhmm>

EDRMA

Selected Applications

VBA.PKInboxInspector
 interface: GUI / VBA
 data: MAPI
 inbox sorting, folder management application

VBA.PKResponseProcessor
 interface: GUI / VBA
 data: MAPI
 response review and report preparation application

PL.AEprocOrder()
 interface: commandline
 data: ADO 2.0
 process raw "order e-mails" by TCID
 uses cf file rule sets for document preprocessing and data element parsing tied to CID; other rules, hardcoded
 generates:
 Acceptable output for review, annotated
 Exceptions, annotated
 Truncated BODYs, annotated
 Raw fields output, annotated
 Raw fields exceptions, annotated
 Rule-eval log (exhaustive)

PL.AEprocBatPrep()
 interface: commandline
 data: ADO 2.0
 takes selected output from PL.AEprocOrder() and transforms to a batch transfer specification via a field map and transform rules cf

PL.AEprocUpdateBatVerbind()
 TBD

PL.AEscrubNormalCanon()
 << not used on the response side >>
 interface: commandline
 data: file handles
 scrubber, normalizer, canonicalizer
 also generates scrambled (non-predictable) row ids and flags AOLs

EDRMA Process Stages [0.5]

Preliminary ECS hooks anticipated

Preprocess Stage

- inspect Inbox using VBA.PKInboxInspector
 - auto-creates folder structure (previous slide)
 - facilitates auto-sort of items into ORDERS, UNSUBS, UNDELS, HARDS, SOFTS, ADDCHANGE, etc.
 - facilitates manu-sort of items into AC, and exception
- report preparation by TCID via AEprocOrder()
 - produces: OUT_report, EXC_report && EXC_BODY diagnostic report

Processing Stage using VBA.PKResponseProcessor

- for each EXC_
 - inspect, correct, reconcile (via Mailing Table lookups) and commit
- for each OUT_
 - inspect and commit to report (or not)
- Acceptable UNION of EXC_ && OUT_ -> Reporting, Update Stages (AEprocBatPrep() || AEprocUpdateBatVerbind())
- Hard exceptions are tagged as such and become Followup RFC's to get additional (critical information)
- FUP (FUPID -> RFC) tags affect routing; tie-back to HARD exception row in this EXC_ report for closure

Response Follow-up Confirmation Stage

- for each non-BOUNCE (ORDERS, UNSUBS, etc.), respond to respondents with an appropriate "confirmation of receipt/action taken" message
- using relevant response template store (templateID)
- this class of response likewise tagged for routing (FUPID -> confirm)

Reporting Stage

- produce order report
 - deliver via fax
 - post to client private web application
 - deliver as formatted batch

Update Stage

- deliver formatted batch to (Verbind, Database) -> sp_???, SQL Executive

*96001

APPENDIX D

Development of the program is the property of the United States Government and is not to be distributed outside the government.

Printed at 10:48:06 on Feb 1999

```

1 # procorder.pl (c) 1998, 1999 E-Dialog, Inc. All rights Reserved.
2 #
3 # PURPOSE DISCUSSION
4 #
5 # Program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
6 # then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
7 #
8 # cf files are campaign optimized:
9 #
10 #   mailingLayout.cf          -> field mappings for a campaign's master mailing table
11 #   messageBodyVotatLayout.cf -> OSNs, field mappings, critical rule designations, order-designation bounds, synonym table
12 #   messageBodyIndicateAddressChange.cf -> rulesets for indicating ADDRESS CHANGES
13 #   messageBodyIndicateBuyAll.cf -> rulesets for indicating BUY ALL state
14 #   messageBodyPreprocessRules.cf -> body text preprocessor configuration and rulesets
15 #
16 # Program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
17 # preparation of those output files, which include:
18 #
19 #   <$ARGV[1]>          -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
20 #   EXC_<...>         -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
21 #   OUT_<...>         -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
22 #   EXC_DETAIL_<...> -> (companion) ..exceptions..discrete fields, etc.
23 #   EXC_BODY_<...>  -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions man
24 #
25 # ually
26 #
27 # output files are (default) tab-delimited text
28 #
29 # PROGRAM REQUIRES
30 #
31 #   perl 5.003 interpreter later compiled for WIN32
32 #   MS ADO components 1.5 (2.0+ recommended)
33 #
34 # RELEASE HISTORY
35 #
36 # 981020AE: 0.5 first usage: to extract carol10 order reports
37 # 981204AE: 0.9 upgrade to handle carol12, multiple item selections
38 # 981211AE: 1.1 improve flagging
39 # 981211AE: 1.5 adapt to extract IDEAS@WORK add/changes
40 # 981222AE: 2.3.1 fix bugs in EXC_BODY reporting / enhance it
41 # 981222AE: 2.4 add CFPATH support
42 # 981223AE: 2.4.1 CFPATH logging; fix synonym (SUBJECT TOKEN) lookup <trailing \>*
43 # 981223AE: 2.5 reorder reporting fields
44 # 981224AE: 2.5.1 update SHIPPING_ADDRESS fields to reflect presence of T.B & s diagnostic fields
45 # 981224AE: 2.5.2 code review/don; update cf to require MAILINGID
46 # 981224AE: 2.6 add support for temporary mailing addresses
47 # 990108AE: 2.6.5 add record-recovery via ADO -> mailing database lookups
48 # 990129AE: 2.7 record-update ADO -> mailing database
49 # 990202AE: 3.0.1 fix leading digits bug in getdatetimeStamp fcn
50 # 990202AE: 3.0.2 fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
51 # 990203AE: 3.1 fix address3 bug; provide regression compatibility for /ADDCHANGE
52 # 990205AE: 3.1.1 throw .notok as a HARD_EXC
53 # 990206AE: 3.1.2
54 #
55 # ##### INCL #####
56 #
57 # use Win32::OLE;
58 #
59 # ##### ENV #####
60 # $crequiredarc = 4;
61 # $grevision = 3.1.2;
62 #
63 # if ($scalar($ARGV) < $crequiredarc) { die "($grevision) usage: procorder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /mult | /add
64 #   change ]\n"; }
65 #
66 # $user = $ARGV[0];

```


E:\Development\NPR\PRJ_hbp\ext\etoider-WORKING\prorder.pl
 Printed at 18:48 on 08 Feb 1999

```

136 C_PRIORITY
137 e_bfirst
138 f_bmiddle
139 g_blast
140 h_btitle
141 i_bcompany
142 j_bdepartment
143 k_baddress1
144 l_baddress2
145 m_baddress3
146 n_bcity
147 o_bstate
148 p_bpostal
149 q_bcountry
150 r_bphone
151 s_bfax
152 t_bemail
153
154 ue_sfirst
155 ue_smiddle
156 ue_slast
157 uh_stitle
158 ui_scompany
159 uj_sdepartment
160 uk_saddress1
161 ul_saddress2
162 um_saddress3
163 un_scity
164 uo_sstate
165 up_spostal
166 uq_scountry
167 ur_sphone
168 us_sfax
169 ut_semail
170
171 ux_from_address
172 uy_location_buy_token
173 v_subject
174 z_fup_required
175 zz_review_type
176
177 );
178
179 ## exceptionsfile - BODY only
180
181 Xgexceptionsbodyfilerrecord = (
182
183 ----->>"{thislogkey, "<<"',
184 ab_review_type
185 ab_except_flags
186 b_rid
187 u_from_address
188 v_subject
189 ----->>"'),
190
191 w_body_below
192
193 );
194
195 ## activitylogfile
196
197 Xgactivitylogrecord = (
198
199 a_lid
200 b_user
201 c_datestamp
202 e_rid
203 f_clientname
204 g_eventtype
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Page 4

FA_Development\PRJ\PRJ_hbap_order\order-WORLD\SGProcOrder.pl
 Printed at 19:46 on 06 Feb 1999

```

203 h_eventDesc => 'e_seventoeSc';
204 z_fup_reqired => 'e_seventoeSc';
205 );
206
207 ##### DATA #####
208 $getkeyPos_Outfile($outFilterRecord);
209 # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)
210 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
211 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
212 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
213 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
214 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
215 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
216 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
217 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
218 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
219 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
220 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
221 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
222 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
223 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
224 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
225 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
226 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
227 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
228 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
229 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
230 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
231 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
232 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
233 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
234 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
235 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
236 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
237 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
238 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
239 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
240 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
241 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
242 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
243 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
244 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
245 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
246 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
247 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
248 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
249 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
250 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
251 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
252 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
253 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
254 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
255 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
256 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
257 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
258 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
259 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
260 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
261 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
262 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
263 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
264 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
265 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
266 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
267 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
268 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
269 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
270 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
271 #print "---- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";

```

F:\Development\PR\PRJ_hbsp_extract\Order-WORKING\prcorder.pl
Printed at 19:48 on 06 Feb 1999

[illegible]

File: Development\PR\PRJ_hbsp_extractOrder-WORKING.prcOrder.pl
 printed at 19:48 on 06 Feb 1999

46

Developmental
 10/10/01 01:00:00

Developmental
 10/10/01 01:00:00

```

525 )
526 )
527 ## A MATCH WAS FOUND, THIS KEY ... push into results set
528 if ($MATCH_DATA_ON) {
529   $countfields++;
530   $matchelement = $results{$key} -> s/(\w+\.|\w+\.|\w+\.|\w+\.)/$space/g;
531   $results{$key} = $matchelement;
532 }
533 ## print "----DEBUG-- result = $results{$key} \n";
534 ## build shipping and billing address blocks
535 CASE: {
536   if ($key == /u[a-z]\./) {
537     ## CONSTRUCT SHIPPING ADDRESSBLOCK for report
538     if ($key == /hijklm\./) { $thisshippingaddressblock .= (length($results{$key}) > 0)?($results{$key}).$space:'';
539     if ($key == /efgnop\./) { $thisshippingaddressblock .= (length($results{$key}) > 0)?($results{$key}).$space:'';
540   }
541   } last CASE;
542   if ($key == /Au\./) {
543     ## CONSTRUCT BILLING ADDRESSBLOCK for report
544     if ($key == /hijklm\./) { $thisbillingaddressblock .= (length($results{$key}) > 0)?($results{$key}).$space:'';
545     if ($key == /efgnop\./) { $thisbillingaddressblock .= (length($results{$key}) > 0)?($results{$key}).$space:'';
546   }
547   } last CASE;
548   }
549   }
550   }
551   }
552   }
553   }
554   }
555   }
556   }
557   }
558   }
559   }
560   }
561   }
562   }
563   }
564   }
565   }
566   }
567   }
568   }
569   }
570   }
571   }
572   }
573   }
574   }
575   }
576   }
577   }
578   }
579   }
580   }
581   }
582   }
583   }
584   }
585   }
586   }

```

Page 10

Development of a program to extract information from a file
 dated 11/15/98 at 15:48:00 by [redacted]

```

587 if ($key == /{SLAYOUT('criticalfields')}}{
588     RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
589     $SEVENTH_RECORD_FUP = ((SeventeenthRequiresFUP == /badad/)?'':'.badad'); $FUPCOUNT++; $EXCPCOUNT++; $seven
590     tDesc = " \<" $thisRecordID. "\> UNPARSABLE STANDARD FIELD NAME=FIELD_VALUE PAIR IS CRITICAL: { $thisLayOutFieldName}"; &logEvent("\*FH_log, seventype,
591     seventDESC, $ouser, $LGOBUG_FTN);
592 }
593
594 }
595
596 #####
597 ### END INNER LOOP ###
598 #####
599 #####
600 #####
601 ### SUMMARY RULES ###
602 #####
603
604 ## throw EX_LISTNO/CUSTNO error exception if address is not OK && iPhone, depending on CID_LID_REQUIRED from cf file
605
606 if ($SLAYOUT('CID_LID_REQUIRED')) {
607     if ($LISTCID_EXCEPT_ON) {
608         RECORD_VALID = 0;
609         $seventeenthRequiresFUP = ((SeventeenthRequiresFUP == /chxclid/)?'':'.chxclid'); $FUPCOUNT++; $EXCPCOUNT++; $seventDESC
610         c = " \<" $thisRecordID. "\> EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent("\*FH_log, seventype, seventDESC, $ouser, $LGOBUG_FTN);
611     }
612 }
613
614 }
615
616 if ($SLAYOUT('CID_LID_REQUIRED')) {
617     if ($LISTCID_EXCEPT_ON && ($ADDRESS_ADEQUATE && $results('\_phone') != /\s+?/)) { $RECORD_VALID = 0; }
618 }
619
620 }
621
622 ## throw EX_MAILINGID exception if cf file says it's required
623
624 if ($SLAYOUT('MAILINGID_REQUIRED')) {
625     if ($MAILINGID_EXCEPT_ON) {
626         RECORD_VALID = 0;
627         $seventeenthRequiresFUP = ((SeventeenthRequiresFUP == /chxmid/)?'':'.chxmid'); $FUPCOUNT++; $EXCPCOUNT++; $seventDESC
628         = " \<" $thisRecordID. "\> MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent("\*FH_log, seventype, seventDESC, $ouser, $LGOBUG_FTN);
629     }
630 }
631
632 }
633
634 }
635
636 ## throw INFO if less than number required
637
638 if (($countfields < $numberRequiredfields) && $thisCUSTNO < 1) {
639     $seventype = 'INFO'; $seventDESC = " \<" $thisRecordID. "\> (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countfields ($numberRequiredfields)"
640     ; &logEvent("\*FH_log, seventype, seventDESC, $ouser, $LGOBUG_FTN);
641 }
642
643 ## tag duplicates for FUP
644
645 if ($dupmap[$mailincnt] == /\0\0\0\0\0\0/) { $seventeenthRequiresFUP = /dupad/; $FUPCOUNT++; }
646
647 ## catch thisbodytop and thisbodybottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)
648
649 if ($RECORD_VALID && $thisbodytop != /\s+?/ && $thisbodybottom != /\s+?/) {
650

```

Development
 10/10/01 10:06:00

Development
 10/10/01 10:06:00

```

651 $RECORD_VALID = 0;
652 $eventtype = 'EX_ERR'; $eventreqresfup = '.chkbody'; $fupcount++; $eventdesc = " \<".$thisrecordid."> CUSTOMER RESPONSE (Top,
653 Bottom) NOT FOUND WHEN EXPECTED"; $logevent("\FH_log, Seventype, $eventdesc, $user, $LGDBUG_FTN);
654 )
655 #####
656 ##### RECORD INVALID: Attempt recovery from mailing database #####
657 #####
658 #####
659 $RECORD_RECOVERED = 0;
660
661 if ($RECORD_VALID && $ADD_CHANGE_ON) {
662   $eventtype = 'INFO'; $eventdesc = " \<".$thisrecordid."> >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"thisma
663   flingdsn\" -> $LAYOUT('thismailingtable'); $logevent("\FH_log, Seventype, $eventdesc, $user, $LGDBUG_FTN);
664
665   $recoveredresults = undef;
666   $recoveredresults = &queryselect_db("\$LAYOUT, 'EMAIL', $thisfromaddress, $LGDBUG_FTN);
667   ## foreach(keys($recoveredresults)) { print "debug== key = $ _ value = $recoveredresults{$ _} \n"; }
668   CASE: {
669     if ($recoveredresults{'_exit_'} < 1) {
670       $RECORD_VALID = 0;
671       $eventtype = 'EX_ERR'; $eventreqresfup = "(/badmob/)?"; $fupcount++; $eventdesc = $eventdesc; $eventdesc
672       = " \<".$thisrecordid."> MAILING TABLE RECOVERY-LOOKUP FAILED ON \"thisfromaddress\"; $logevent("\FH_log, Seventype, $eventdesc, $user, $LGDBUG_F
673       TN);
674     }
675     last CASE;
676   }
677   if ($recoveredresults{'_exit_'} == 1) {
678     $RECORD_RECOVERED = 1;
679     $RECORD_VALID = 1;
680     $eventtype = 'MKT_FUP'; $eventreqresfup = "(/goodmob/)?"; $fupcount++; $eventdesc = $eventdesc; $eventdesc
681     = " \<".$thisrecordid."> <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"thisfromaddress\"; $logevent("\FH_log, Seventype, $eventdesc, $user, $LGDBUG_F
682     TN);
683   }
684   SC = " \<".$thisrecordid.">
685   ## construct $thismobbillingaddress
686   foreach ($key(sort(keys($MOB_LAYOUT))) {
687     $value = $MOB_LAYOUT{$key};
688     ## print "debug key = $key value = $value \n";
689     if ($key =~ /([a-zA-Z0-9_@])/) { $thismobbillingaddressblock .= (length($recoveredresults{$value}) > 0)?($recoveredresults{$va
690     lue}).$pipe.'':; }
691     if ($key =~ /([a-zA-Z0-9_@])/) { $thismobbillingaddressblock .= (length($recoveredresults{$value}) > 0)?($recoveredresults{$va
692     lue}).$space.'':; }
693   }
694   ## print "debug mailingid = $recoveredresults{'ED_MAILINGID'} - custno = $recoveredresults{'CID'} - current custno = $thiscustno\n";
695   if ($thismailingid =~ /\s+/) { $thismailingid = -1 } { $thismailingid = $recoveredresults{'ED_MAILINGID'}; }
696   if ($thiscustno =~ /\s+/) { $thiscustno = -1 } { $thiscustno = $recoveredresults{'CID'}; }
697   last CASE;
698 }
699
700
701
702
703
704
705
706
707
708
709
710
711

```


Page 13

EX: Development\PRJ\PRJ_heap\src\order-WORKING\pr_order.c
 Printed at: 18:48 on 06 Feb 1999

```

772 )
773
774
775 ### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report
776
777 unless($ADDRESS_CHANGE) { if($BUYTokenPosition eq 'none') { $RECORD_VALID = 0; $EO_ORDER_VALUE = 2; } }
778
779 if($RECORD_VALID) {
780     $reportLineCount++;
781     ## update MAILING TABLE
782
783     ## write out files
784
785     $LGDEBUG_FTN: &writeRecord(\*FH_out,$gfieldDelimiter,($thiskey-&getNextDbkey_return($goutFileDbkeysFileName,$LGDEBUG_FTN)),\XgoutFileRecord,'EXTENDED',
786     RMAL', $LGDEBUG_FTN);
787     &writeRecord(\*FH_report,$gfieldDelimiter,($thiskey-&getNextDbkey_return($greportFileDbkeysFileName,$LGDEBUG_FTN)),\XgreportFileRecord,'NO
788
789     ## include ALL BODYS in the EXC_BODY file for potential review
790     if($eventRequiresFUP == /\s+7/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }
791
792     &writeRecord(\*FH_excBODYreport,$gfieldDelimiter,$thisbodylogkey-&getNextDbkey_return($gexceptionsBODYDbkeysFileName,$LGDEBUG_FTN)
793     ,\XgexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
794     $outFileCount++;$reportFileCount++;
795 }
796
797 #####
798 ## write exception record data to outfiles ##
799 #####
800
801 if(!($RECORD_VALID) {
802     $EXCEPTION_TYPE = 'HARD-EXC';
803     &writeRecord(\*FH_exc,$gfieldDelimiter,$thiskey-&getNextDbkey_return($gexceptionsDBkeysFileName,$LGDEBUG_FTN),\XgoutFileRecord,'EXTENDED',
804     RMAL', $LGDEBUG_FTN);
805     &writeRecord(\*FH_excBODYreport,$gfieldDelimiter,$thiskey-&getNextDbkey_return($gexceptionsBODYDbkeysFileName,$LGDEBUG_FTN),\XgreportFileRecord,'NO
806     ptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
807     $exceptionsFileCount++; $excReportFileCount++;
808 }
809
810 print "\n";
811 }
812
813 #####
814 ## end main loop #####
815
816 ## print STDOUT exit state
817
818 $dateNow = &getdatestamp('WIN', $LGDEBUG_FTN);
819
820 print "\n*** [ProcOrder version $grevision] Done! $totalRecords records processed by $user/$gcPath for client: $gclientName *\\n\\n";
821
822 print "SUMMARY REPORT for: ".$dateNow."\\n\\n";
823
824 print "FILES WRITTEN:\\n";
825
826 print "  \"X100\" $reportFileCount;
827
828 print "  \"X100\" $excReportFileCount;
829
830 print "  \"X100\" $outFileCount;
831
832 print "  \"X100\" $reportFileCount;
833
834 print "  \"X100\" $excReportFileCount;
835
836 print "  \"X100\" $outFileCount;

```

```

835 print " records written to OUTPUT FILE (all fields): $outfile";
836 printf "\n%ld", $exceptionsfilecount;
837 print " records written to EXCEPTIONS FILE (all fields): $exceptionsfile";
838 print "\nLogging\n";
839 print "All rule failure events for all time logged in detail to: $activitylogfile\n";
840 printf "\n%ld", $totalrecords;
841 print " records for this session are keyed, categorized in: $excbodyreportfile, for review in an editor\n";
842
843 $eventtype = "X-INFO"; $eventdesc = "ProcOrder R $gcrevision EXIT OK - USER: $guser/$gcpath CLIENT: $gcclientname
844 $logging, $eventtype, $eventdesc, $guser, $lgdebug_fth);
845 close($fh_out); close($fh_exc); close($fh_log); close($fh_report); close($fh_excbodyreport);
846
847 exit;
848
849 #####
850 ##### SUBS #####
851 #####
852
853 # -----
854 #
855 #
856
857 sub alphaordered {
858     my($singleElementArray) = @_;
859     my($singleElementArray) = @_;
860     my $alpha = 0;
861     my $i = 0;
862     my $j = 0;
863     my $base = $singleElementArray[0];
864     for($i = 1; $i <= $singleElementArray; $i++) {
865         $base = ord($base); $ordarr = ord($singleElementArray[$i]);
866         if($base > $ordarr) { return $alpha; }
867         $base = $singleElementArray[$i];
868     }
869     return $alpha;
870 }
871
872 if($ordbase > $ordarr) { return $alpha; }
873 $base = $singleElementArray[$i];
874 }
875
876 $alpha = 1;
877
878 return $alpha;
879
880
881 }
882
883 # -----
884 #
885 #
886 sub canonicalize {
887     my($inRecord) = @_;
888     my($fieldindex, $rec, $buf, $field) = undef;
889     my($field) = undef;
890     my $space = " ";
891     my $d_escaped = "d_escaped";
892     my $d_escaped = "d_escaped";
893     my $inRecord = uc($inRecord);
894     my $allfields = undef;
895     my $allfields = undef;
896     my $allfields = split(/$fielddelimiter/, $inRecord);
897     my $fieldindex = undef;
898     for($fieldindex=0; $fieldindex <= $#allfields; $fieldindex++) {

```

Page 15

F:\Development\PR_APP\BBA\ExtOrder-WORKING\ExtOrder.pl
 Printed at 9:48 on 06 Feb 1999

```

902 if ($fieldsIndex != $smallfieldpos && $fieldsIndex != $fromaddressfieldpos) {
903   if ($fieldsIndex == $companyfieldpos || $fieldsIndex == $lastnamefieldpos || $fieldsIndex == $title
904     efieldpos || $fieldsIndex == $countryfieldpos || $fieldsIndex == $departmentfieldpos { $allfields[$fieldsIndex] == s/
905     -/dashescape/g; }
906   $allfields[$fieldsIndex] == s/\s+//;
907   $allfields[$fieldsIndex] == s/\s+//;
908   $allfields[$fieldsIndex] == s/\s+//;
909   $allfields[$fieldsIndex] == s/\s+//;
910   $allfields[$fieldsIndex] == s/\s+//;
911   $allfields[$fieldsIndex] == s/\s+//;
912   $allfields[$fieldsIndex] == s/\s+//;
913   $allfields[$fieldsIndex] == s/\s+//;
914   $allfields[$fieldsIndex] == s/\s+//;
915   $allfields[$fieldsIndex] == s/\s+//;
916   $allfields[$fieldsIndex] == s/\s+//;
917   $allfields[$fieldsIndex] == s/\s+//;
918   $allfields[$fieldsIndex] == s/\s+//;
919   $allfields[$fieldsIndex] == s/\s+//;
920   $allfields[$fieldsIndex] == s/\s+//;
921   $allfields[$fieldsIndex] == s/\s+//;
922   $allfields[$fieldsIndex] == s/\s+//;
923   $allfields[$fieldsIndex] == s/\s+//;
924   $allfields[$fieldsIndex] == s/\s+//;
925   $allfields[$fieldsIndex] == s/\s+//;
926   $allfields[$fieldsIndex] == s/\s+//;
927   $allfields[$fieldsIndex] == s/\s+//;
928   $allfields[$fieldsIndex] == s/\s+//;
929   $allfields[$fieldsIndex] == s/\s+//;
930   $allfields[$fieldsIndex] == s/\s+//;
931   $allfields[$fieldsIndex] == s/\s+//;
932   $allfields[$fieldsIndex] == s/\s+//;
933   $allfields[$fieldsIndex] == s/\s+//;
934   $allfields[$fieldsIndex] == s/\s+//;
935   $allfields[$fieldsIndex] == s/\s+//;
936   $allfields[$fieldsIndex] == s/\s+//;
937   $allfields[$fieldsIndex] == s/\s+//;
938   $allfields[$fieldsIndex] == s/\s+//;
939   $allfields[$fieldsIndex] == s/\s+//;
940   $allfields[$fieldsIndex] == s/\s+//;
941   $allfields[$fieldsIndex] == s/\s+//;
942   $allfields[$fieldsIndex] == s/\s+//;
943   $allfields[$fieldsIndex] == s/\s+//;
944   $allfields[$fieldsIndex] == s/\s+//;
945   $allfields[$fieldsIndex] == s/\s+//;
946   $allfields[$fieldsIndex] == s/\s+//;
947   $allfields[$fieldsIndex] == s/\s+//;
948   $allfields[$fieldsIndex] == s/\s+//;
949   $allfields[$fieldsIndex] == s/\s+//;
950   $allfields[$fieldsIndex] == s/\s+//;
951   $allfields[$fieldsIndex] == s/\s+//;
952   $allfields[$fieldsIndex] == s/\s+//;
953   $allfields[$fieldsIndex] == s/\s+//;
954   $allfields[$fieldsIndex] == s/\s+//;
955   $allfields[$fieldsIndex] == s/\s+//;
956   $allfields[$fieldsIndex] == s/\s+//;
957   $allfields[$fieldsIndex] == s/\s+//;
958   $allfields[$fieldsIndex] == s/\s+//;
959   $allfields[$fieldsIndex] == s/\s+//;
960   $allfields[$fieldsIndex] == s/\s+//;
961   $allfields[$fieldsIndex] == s/\s+//;
962   $allfields[$fieldsIndex] == s/\s+//;

```


T:\Development\PHAPP\Bmap_ExtractOrder-WORKING\BmapOrder.pl
 Printed at 19:48 on 06 Feb 1999

```

963 # -----
964 #
965 #
966 sub checkdupreturnmap (
967   my($perioHDB,$GDEBUG) = @_;
968   my($pos) = 0;
969   my($pos) = 0;
970   ## prepare map of duplicates on FromAddress
971   my %dupmap;
972   my %dupmap;
973   my %dupmap;
974   my %dupmap;
975   my %dupmap;
976   my %dupmap;
977   for $j (0 .. $#DB) {
978     $dupmap{$j} = $DB[$j]{'FromAddress'}
979   }
980   ##
981   ##
982   ##
983   my @mirroredupmap;
984   my %key;
985   my %value;
986   my %arr;
987   my %number;
988   my %number;
989   my %number;
990   foreach $key(keys %dupmap) {
991     $value = $dupmap{$key};
992     next if($value =~ /\S+?/);
993     @arr = undef;
994     @arr = grep /$value/, @mirroredupmap;
995     $number = scalar(@arr);
996     if($number > 1) {
997       9. Seventype, $eventdesc, $user, $GDEBUG_FTN;
998       $dupmap{$key} = ("###", $number, "###", $dupmap{$key});
999       ## print $dupmap{$key} . "\n";
1000       ## print "key=" . $key . " value=" . $dupmap{$key} . " occurrences=" . $number . "\n";
1001     }
1002   }
1003   return %dupmap;
1004 }
1005
1006 sub checkmultiruy_Extract (
1007   my($thisline,$thismailingID,$pos,$perLAYOUT,$GDEBUG) = @_;
1008   my($exit) = 0;
1009   my($exit) = 0;
1010   my($exit) = 0;
1011   my($exit) = 0;
1012   my($exit) = 0;
1013   my($exit) = 0;
1014   my($exit) = 0;
1015   my($exit) = 0;
1016   my($exit) = 0;
1017   my($exit) = 0;
1018   my($exit) = 0;
1019   my($exit) = 0;
1020   my($exit) = 0;
1021   my($exit) = 0;
1022   my($exit) = 0;
1023   my($exit) = 0;
1024   my($exit) = 0;
1025   my($exit) = 0;
1026   my($exit) = 0;
1027   my($exit) = 0;
1028   my($exit) = 0;
1029   my($exit) = 0;
1030   my($exit) = 0;

```

[illegible]

```

1100 }
1101 #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {
1102 #
1103 # print "-- debug -- REJECT - range case $match| out of alpha order with current stack!\n";
1104 # if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1105 # SPUSH = 0;
1106 #
1107 #
1108 #
1109 #
1110 if(defined($match) && $match =~ /\$LAYOUT{($thisCheckedMailID)}\/) {
1111 ## print "--debug-- REJECT - not in RANGE = $LAYOUT{($thisCheckedMailID)} match $match| in line $thisPreExtractionLine \n";
1112 # if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1113 # SPUSH = 0;
1114 #
1115 #
1116 #
1117 #
1118 #
1119 #
1120 #
1121 #
1122 #
1123 #
1124 #
1125 #
1126 #
1127 #
1128 #
1129 #
1130 #
1131 #
1132 #
1133 #
1134 #
1135 #
1136 #
1137 #
1138 #
1139 #
1140 #
1141 #
1142 #
1143 #
1144 #
1145 #
1146 #
1147 #
1148 #
1149 #
1150 #
1151 #
1152 #
1153 #
1154 #
1155 #
1156 #
1157 #
1158 #
1159 #
1160 #
1161 #
1162 #
1163 #
1164 #
1165 #
1166 #
1167 #
1168 #
1169 #
1170 #
1171 #
1172 #
1173 #
1174 #
1175 #
1176 #
1177 #
1178 #
1179 #
1180 #
1181 #
1182 #
1183 #
1184 #
1185 #
1186 #
1187 #
1188 #
1189 #
1190 #
1191 #
1192 #
1193 #
1194 #
1195 #
1196 #
1197 #
1198 #
1199 #
1200 #
1201 #
1202 #
1203 #
1204 #
1205 #
1206 #
1207 #
1208 #
1209 #
1210 #
1211 #
1212 #
1213 #
1214 #
1215 #
1216 #
1217 #
1218 #
1219 #
1220 #
1221 #
1222 #
1223 #
1224 #
1225 #
1226 #
1227 #
1228 #
1229 #
1230 #
1231 #
1232 #
1233 #
1234 #
1235 #
1236 #
1237 #
1238 #
1239 #
1240 #
1241 #
1242 #
1243 #
1244 #
1245 #
1246 #
1247 #
1248 #
1249 #
1250 #
1251 #
1252 #
1253 #
1254 #
1255 #
1256 #
1257 #
1258 #
1259 #
1260 #
1261 #
1262 #
1263 #
1264 #
1265 #
1266 #
1267 #
1268 #
1269 #
1270 #
1271 #
1272 #
1273 #
1274 #
1275 #
1276 #
1277 #
1278 #
1279 #
1280 #
1281 #
1282 #
1283 #
1284 #
1285 #
1286 #
1287 #
1288 #
1289 #
1290 #
1291 #
1292 #
1293 #
1294 #
1295 #
1296 #
1297 #
1298 #
1299 #
1300 #
1301 #
1302 #
1303 #
1304 #
1305 #
1306 #
1307 #
1308 #
1309 #
1310 #
1311 #
1312 #
1313 #
1314 #
1315 #
1316 #
1317 #
1318 #
1319 #
1320 #
1321 #
1322 #
1323 #
1324 #
1325 #
1326 #
1327 #
1328 #
1329 #
1330 #
1331 #
1332 #
1333 #
1334 #
1335 #
1336 #
1337 #
1338 #
1339 #
1340 #
1341 #
1342 #
1343 #
1344 #
1345 #
1346 #
1347 #
1348 #
1349 #
1350 #
1351 #
1352 #
1353 #
1354 #
1355 #
1356 #
1357 #
1358 #
1359 #
1360 #
1361 #
1362 #
1363 #
1364 #
1365 #
1366 #
1367 #
1368 #
1369 #
1370 #
1371 #
1372 #
1373 #
1374 #
1375 #
1376 #
1377 #
1378 #
1379 #
1380 #
1381 #
1382 #
1383 #
1384 #
1385 #
1386 #
1387 #
1388 #
1389 #
1390 #
1391 #
1392 #
1393 #
1394 #
1395 #
1396 #
1397 #
1398 #
1399 #
1400 #
1401 #
1402 #
1403 #
1404 #
1405 #
1406 #
1407 #
1408 #
1409 #
1410 #
1411 #
1412 #
1413 #
1414 #
1415 #
1416 #
1417 #
1418 #
1419 #
1420 #
1421 #
1422 #
1423 #
1424 #
1425 #
1426 #
1427 #
1428 #
1429 #
1430 #
1431 #
1432 #
1433 #
1434 #
1435 #
1436 #
1437 #
1438 #
1439 #
1440 #
1441 #
1442 #
1443 #
1444 #
1445 #
1446 #
1447 #
1448 #
1449 #
1450 #
1451 #
1452 #
1453 #
1454 #
1455 #
1456 #
1457 #
1458 #
1459 #
1460 #
1461 #
1462 #
1463 #
1464 #
1465 #
1466 #
1467 #
1468 #
1469 #
1470 #
1471 #
1472 #
1473 #
1474 #
1475 #
1476 #
1477 #
1478 #
1479 #
1480 #
1481 #
1482 #
1483 #
1484 #
1485 #
1486 #
1487 #
1488 #
1489 #
1490 #
1491 #
1492 #
1493 #
1494 #
1495 #
1496 #
1497 #
1498 #
1499 #
1500 #
1501 #
1502 #
1503 #
1504 #
1505 #
1506 #
1507 #
1508 #
1509 #
1510 #
1511 #
1512 #
1513 #
1514 #
1515 #
1516 #
1517 #
1518 #
1519 #
1520 #
1521 #
1522 #
1523 #
1524 #
1525 #
1526 #
1527 #
1528 #
1529 #
1530 #
1531 #
1532 #
1533 #
1534 #
1535 #
1536 #
1537 #
1538 #
1539 #
1540 #
1541 #
1542 #
1543 #
1544 #
1545 #
1546 #
1547 #
1548 #
1549 #
1550 #
1551 #
1552 #
1553 #
1554 #
1555 #
1556 #
1557 #
1558 #
1559 #
1560 #
1561 #
1562 #
1563 #
1564 #
1565 #
1566 #
1567 #
1568 #
1569 #
1570 #
1571 #
1572 #
1573 #
1574 #
1575 #
1576 #
1577 #
1578 #
1579 #
1580 #
1581 #
1582 #
1583 #
1584 #
1585 #
1586 #
1587 #
1588 #
1589 #
1590 #
1591 #
1592 #
1593 #
1594 #
1595 #
1596 #
1597 #
1598 #
1599 #
1600 #
1601 #
1602 #
1603 #
1604 #
1605 #
1606 #
1607 #
1608 #
1609 #
1610 #
1611 #
1612 #
1613 #
1614 #
1615 #
1616 #
1617 #
1618 #
1619 #
1620 #
1621 #
1622 #
1623 #
1624 #
1625 #
1626 #
1627 #
1628 #
1629 #
1630 #
1631 #
1632 #
1633 #
1634 #
1635 #
1636 #
1637 #
1638 #
1639 #
1640 #
1641 #
1642 #
1643 #
1644 #
1645 #
1646 #
1647 #
1648 #
1649 #
1650 #
1651 #
1652 #
1653 #
1654 #
1655 #
1656 #
1657 #
1658 #
1659 #
1660 #
1661 #
1662 #
1663 #
1664 #
1665 #
1666 #
1667 #
1668 #
1669 #
1670 #
1671 #
1672 #
1673 #
1674 #
1675 #
1676 #
1677 #
1678 #
1679 #
1680 #
1681 #
1682 #
1683 #
1684 #
1685 #
1686 #
1687 #
1688 #
1689 #
1690 #
1691 #
1692 #
1693 #
1694 #
1695 #
1696 #
1697 #
1698 #
1699 #
1700 #
1701 #
1702 #
1703 #
1704 #
1705 #
1706 #
1707 #
1708 #
1709 #
1710 #
1711 #
1712 #
1713 #
1714 #
1715 #
1716 #
1717 #
1718 #
1719 #
1720 #
1721 #
1722 #
1723 #
1724 #
1725 #
1726 #
1727 #
1728 #
1729 #
1730 #
1731 #
1732 #
1733 #
1734 #
1735 #
1736 #
1737 #
1738 #
1739 #
1740 #
1741 #
1742 #
1743 #
1744 #
1745 #
1746 #
1747 #
1748 #
1749 #
1750 #
1751 #
1752 #
1753 #
1754 #

```

F:\Development\PRJ293\nbsp_extractOrder-WG\MINGW\ProcOrder.pl
printed at 19:28 on 06 Feb 1999

58

Page 20

Printed: 01/18/2006 15:06:33
 File: C:\Program Files\IBM\SPSS\Statistica\11.0\Source\ExtractOrder.mak

```

1226     )
1227
1228
1229     ##print "\n-$thisRecordId- debug singlestack -- \n";
1230     ##print @singlestack;
1231     ##print "\n-- debug testSinglestack -- \n";
1232     ##print @testSinglestack;
1233     ##print "\n-$thisRecordId----- \n";
1234
1235
1236     return $exit;
1237 }
1238
1239
1240 #-----
1241 #
1242 #
1243
1244 sub detectBuyTokens {
1245     my($MULTI_BUY_ON,$SUBJ_BUY_TOKEN,$thisSubject,$thisMailIngID,$thisBodyTop,$thisBodyBottom,$SPRSCALARBottomLines,$GDEBUG)
1246     = @_;
1247
1248     my $LAYOUT
1249     = $SPRSHASHLAYOUT;
1250     my @topLines
1251     = $SPRSCALARTopLines;
1252     my @bottomLines
1253     = $SPRSCALARBottomLines;
1254     my $buyTokenPosition
1255     = 'none';
1256     $thisBuyStatus
1257     = 'undef';
1258     $thisItemsSelected
1259     = 'undef';
1260     $thisItemsSelectedStack
1261     = 'undef';
1262
1263     my $exit = 0;
1264
1265     ## print "--debug-- buyToken= [$SUBJ_BUY_TOKEN]\n";
1266
1267     FIND_BUY_TOKENS: {
1268
1269         ##
1270         ##
1271         ## &checkMultiBuy_Extract returns success on a good match/extraction;
1272         ## @thisItemsSelected is created/populated at this time as well
1273         ##
1274         ## check $thisSubject
1275
1276         ## print "\n\n=====debug-- SUBJECT = [$thisSubject] \n";
1277         if(!($MULTI_BUY_ON && $thisSubject =~ /$SUBJ_BUY_TOKEN/)) {
1278             $buyTokenPosition = 'S'; $thisBuyStatus = ('<'. $buyTokenPosition. '>'. $thisSubject. '!');
1279
1280             ## print "--debug-- thisBuyStatus= [$thisBuyStatus] \n";
1281             last FIND_BUY_TOKENS;
1282         }
1283
1284         if($MULTI_BUY_ON) {
1285             if(&checkMultiBuy_Extract($thisSubject,$thisMailIngID,'S',$LAYOUT,$GDEBUG)) {
1286                 $buyTokenPosition = 'S'; $thisBuyStatus = '<'. $buyTokenPosition. '>';
1287                 last FIND_BUY_TOKENS;
1288             }
1289         }
1290
1291         ## check @topLines
1292         ## print "\n\n=====debug-- thisBodyTop = [$thisBodyTop] \n";
1293

```

Full Developmental Report: hbsp_extractOrder WORKINGPROG09a001
 Printed at 16:48 on 06-Feb-1999

```

1294 if(!MULTI_BUY_ON) {
1295     foreach(@toplines) {
1296         $histopline = $-;
1297         if($histopline =~ /%$subj_buyToken/) {
1298             $buyTokenPosition = 'T';
1299             $thisbuyStatus = 'c'. $buyTokenPosition. '>'. $histopline;
1300             ## print "--debug-- thisbuyStatus= {$thisbuyStatus} \n";
1301         }
1302     }
1303     last FIND_BUY_TOKENS;
1304 }
1305 if($MULTI_BUY_ON) {
1306     if(&checkMultiBuy_Extract($thisbodyTop,$thismailingID,'T',\%LAYOUT,%codebug)) {
1307         $exit = 1;
1308         $buyTokenPosition = 'T';
1309         $thisbuyStatus = 'c'. $buyTokenPosition. '>';
1310         last FIND_BUY_TOKENS;
1311     }
1312 }
1313 ## check @bottomlines
1314 ## print "\n\n=====debug-- thisbodybottom = {$thisbodybottom} \n";
1315 if(!MULTI_BUY_ON) {
1316     foreach(@bottomlines) {
1317         $thisbottomline = $_;
1318         if($thisbottomline =~ /%$subj_buyToken/) {
1319             $buyTokenPosition = 'B';
1320             $thisbuyStatus = 'c'. $buyTokenPosition. '>'. $thisbottomline;
1321             ## print "--debug-- thisbuyStatus= {$thisbuyStatus} \n";
1322         }
1323     }
1324     last FIND_BUY_TOKENS;
1325 }
1326 }
1327 }
1328 }
1329 }
1330 }
1331 }
1332 }
1333 }
1334 }
1335 }
1336 }
1337 }
1338 }
1339 }
1340 }
1341 }
1342 }
1343 }
1344 }
1345 }
1346 }
1347 }
1348 }
1349 }
1350 }
1351 }
1352 }
1353 }
1354 }
1355 }
1356 }
1357 }
1358 }
1359 }
1360 }
1361 }
1362 }

```

```

1363 last FIND_BUY_TOKENS;
1364 }
1365 )
1366 )
1367 )
1368 )
1369 )
1370 ## print "--debug-- ===== $! ===== buyTokenPosition= |$buyTokenPosition|\\";
1371
1372 return $buyTokenPosition;
1373 }
1374 }
1375 }
1376 }
1377 }
1378 }
1379 }
1380 sub extractDuplicatesSelectedStack (
1381     my @thisItemsSelected) = @_;
1382
1383     my $thisItemsSelectedStack = undef;
1384     my $thisItemsSelectedStackSeparator = '.';
1385     my $i = undef;
1386     for($i = 0; $i <= $#thisItemsSelected; $i++) {
1387         my $item = $thisItemsSelected[$i]; # all items must be normalized!
1388         next if($item !~ /\S+/);
1389         $item =~ normalizeSpacing($item);
1390         $thisItemsSelectedStack .= ($item.$thisItemsSelectedStackSeparator);
1391     }
1392 }
1393
1394 $thisItemsSelectedStack =~ s/\.\s+//;
1395
1396 return $thisItemsSelectedStack;
1397 }
1398
1399 }
1400
1401 }
1402 }
1403 }
1404 }
1405 }
1406 }
1407 }
1408 sub getDateStamp {
1409     # get standardized date stamp (UNIX, WINNT)
1410     # returns a date stamp
1411     my($platform,$toDateFunc,$FTW) = @_;
1412     my($splatform,$toDateFunc,$FTW) = @_;
1413     if(!defined($splatform)) { $splatform = 'WIN'; }
1414     my $sTIME = ' ';
1415     my $day = undef;
1416     my $val = undef;
1417     if ($splatform ne 'WIN') {
1418         $val = `date +%Y%m%d%H%M%S-%s`;
1419     }
1420     else {
1421         $val = `date /t`;
1422         chop($val);
1423         $val =~ s/(\d+)(\d+)(\d+)(\d+)(\d+)(\d+)/$1$2$3$4$5$6/;
1424     }
1425     if ($splatform eq 'WIN') {
1426         $sTIME = `echo %time%`;
1427         chop($sTIME);
1428         $sTIME =~ s/(\d+)(\d+)(\d+)(\d+)(\d+)(\d+)/$1$2$3$4$5$6/;
1429     }
1430 }
1431

```

F:\Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```

1432 my $leadingpigit = $1;
1433 if (length($leadingpigit) == 1) { $leadingpigit = ('0' . $leadingpigit); }
1434 $val := $leadingpigit.$2.$3.$4.'.'. $day;
1435 }
1436 if ($LGDBUG_FTN) { print "DEBUG_gdate=" . $val . "\n"; }
1437 return $val;
1438 }
1439 }
1440 }
1441 }
1442 }
1443 }
1444 }
1445 }
1446 }
1447 sub getfieldkeys {
1448 my($ptrLayout) = @_;
1449 my($LAYOUT) = %$ptrLayout;
1450 }
1451 ## prepare and count required fields from body data layout for check against body
1452 my($fieldkeys) = undef;
1453 my($sk) = undef;
1454 ##$f = scalar(@fieldkeys); print "--debug-- scalarStartFtn=$f \n";
1455 foreach $k (sort keys %LAYOUT) {
1456 next if ($k =~ /\[a-z\]\[1,2\]/); ## these elements are not fieldnames but are other CF parameters (name \t value)
1457 ## print "--debug-- k= $k \n";
1458 if ($k =~ /\S+7/) { push(@fieldkeys, $k); }
1459 }
1460 shift @fieldkeys;
1461 ## $f = scalar(@fieldkeys); print "--debug-- scalarEndFtn=$f \n";
1462 return @fieldkeys;
1463 }
1464 }
1465 }
1466 }
1467 }
1468 }
1469 }
1470 }
1471 }
1472 }
1473 }
1474 }
1475 }
1476 sub getKeyPos_Outfile {
1477 my($ptrHASHRecordTemplate) = @_;
1478 my($XHASHRecordTemplate) = %$ptrHASHRecordTemplate;
1479 }
1480 my $record = undef;
1481 }
1482 }
1483 $salutationFieldPos = -1;
1484 $idFieldPos = -1;
1485 $titleFieldPos = -1;
1486 $lastNameFieldPos = -1;
1487 $firstNameFieldPos = -1;
1488 $middleNameFieldPos = -1;
1489 $sexFieldPos = -1;
1490 $stateFieldPos = -1;
1491 $countryFieldPos = -1;
1492 $postalCodeFieldPos = -1;
1493 $emailFieldPos = -1;
1494 $companyFieldPos = -1;
1495 $addressFieldPos = -1;
1496 $departmentFieldPos = -1;
1497 $cityFieldPos = -1;
1498 $fromAddressFieldPos = -1;
1499 }
1500 }

```



```

1568 flock(KEYS, LOCK_EX);
1569
1570     my($count)      = <KEYS>;
1571     $nextkey       = $count+1;
1572     seek(KEYS,0,$POS_EOF);
1573     print KEYS $nextkey;
1574
1575 flock(KEYS, LOCK_UN);
1576 close(KEYS);
1577 return $nextkey;
1578 }
1579
1580 -----
1581 #
1582 #
1583 #
1584
1585 sub HASHloadFieldMap {
1586     my($filename,$LOGDEBUG_FTN) = @_;
1587     my($filename,"$filename") { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \${keyFileName}"; $logEvent("\$FH_log, $eventType, $eventDesc, $user, $LOGDEBUG_FTN); }
1588     $/ = "\n";
1589     $/ = "\n";
1590     my($fieldMapTemp) = <FH>;
1591     my($key) = undef;
1592     my($value) = undef;
1593     my($fieldMap) = undef;
1594     close(FH);
1595     foreach (@fieldMapTemp) {
1596         my $item = $_;
1597         next if (($item =~ /\n/) || ($item =~ /\s+7/)); # comments
1598         ($key, $value) = split(/[\t\,]/,$item);
1599         $value =~ s/[\n\r]//g;
1600         $fieldMap{$key} = $value;
1601     }
1602     return %fieldMap;
1603 }
1604
1605 -----
1606 #
1607 #
1608 #
1609
1610 sub loadFile_SCALAR {
1611     my($filename) = @_;
1612     if ($!open(FILE, "$filename") { $eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open filename: \${filename}"; $logEvent("\$FH_log, $eventType, $eventDesc, $user, $LOGDEBUG_FTN); }
1613     $/ = "\n";
1614     $/ = "\n";
1615     my $buf = $_;
1616     next if ($buf =~ /\s+7/ || $buf =~ /\n\r/);
1617 }

```

FA Development\PRJ\PRJ_1597\ProcOrder-WORKING\ProcOrder.pl
 Printed at 16:48 on 06 Feb 1999

```

1635     push(@all, $buf);
1636 }
1637
1638 return @all;
1639
1640 }
1641 #
1642 # -----
1643 #
1644 #
1645 #
1646 sub logEvent {
1647     my($FH, $eventType, $eventDesc, $user, $LDEBUG_FTH) = @_;
1648     my($FH, $eventType, $eventDesc, $user, $LDEBUG_FTH);
1649     my $datetime = &getdatestamp('WIN', $LDEBUG_FTH);
1650     # print event to STDOUT
1651     # print event to STDOUT
1652     if($eventType eq 'ABORT') {
1653         $foo = "mapisend -u administrator -p password -r aestes\@e-dialog.com -s \"** $datetime procOrder - Critical Error\" -m \"$user $clientName $eventT
1654         ype $eventDesc\"";
1655         die "Error! Cannot log! \"$eventType.\" \"$eventDesc.\" \"\n\";
1656     }
1657     if($eventType !~ /_x_/) { print $eventType.\"\\t\".$eventDesc.\"\\n\"; }
1658     else { $eventType =~ s/_x_/; }
1659     # write event to activity log
1660     $writerRecord(\"*FH_log, $fieldName, $thisLogKey=&getNextDbkey_return($activityLogKeysFileName, $LDEBUG_FTH), $activityLogRecord,\"\", $LDEBUG_F
1661     TH);
1662     if($eventType eq 'FAIL') {
1663         $foo = "mapisend -u administrator -p password -r aestes\@e-dialog.com -s \"** $datetime procOrder - Critical Error\" -m \"$user $clientName
1664         $eventType $eventDesc\"";
1665         exit;
1666     }
1667     $foo = "mapisend -u administrator -p password -r aestes\@e-dialog.com -s \"** $datetime procOrder - Critical Error\" -m \"$user $clientName
1668     $eventType $eventDesc\"";
1669     exit;
1670 }
1671
1672 }
1673
1674 # -----
1675 #
1676 #
1677 #
1678 sub normalizeSpacing {
1679     my($dirty) = @_;
1680     $dirty =~ s/\\s{2,}/space/g; $dirty =~ s/\\s+$/; $dirty =~ s/\\s+//;
1681     $clean = $dirty;
1682     return $clean;
1683 }
1684
1685 # -----
1686 #
1687 #
1688 #
1689 #
1690 #
1691 sub queryUPDATE_DB {
1692     my($SPQLAYOUT, $thisQueryKey, $thisQueryFieldName, $fieldName, $fieldValue, $fieldName1, $fieldValue1, $RECORD_RECOVERED, $LDEBUG_FTH) = @_;
1693     #print "** OSN = $LAYOUT{'thisSmallingDSN'} \\n\";
1694     my($LAYOUT) = $SPQLAYOUT;
1695     my $rs = undef;
1696     my $Conn = undef;
1697     my $Conn = undef;
1698     my $Conn = undef;
1699     my $Conn = undef;
1700     my $Conn = undef;
1701     my $Conn = undef;

```

FA Development\PRJ\PRJ_hisp_extract_order-WORKING\hisp_order.pl
 Printed at 18:48 on 06 Feb 1999

```

1701 my $errors = undef;
1702 my $fieldNames = undef;
1703 my $pid = -1;
1704 my $recCount = 0;
1705 my $EXIT = 1;
1706
1707 # construct/open DSN
1708
1709 my $Conn = undef;
1710 $Conn = new Win32::OLE("ADODB.Connection");
1711 $Conn->open($LAYOUT{'thismailingpsn'}, "", "", 1);
1712
1713 my($DSN) = undef;
1714
1715 # Load data into recordset
1716
1717 my $mailingTable = $LAYOUT{'thismailingTable'};
1718 my $foo = undef;
1719
1720 # look to see if update record exists, $thisQueryKey
1721
1722 unless($RECORD_RECOVERED) {
1723     my $RECORD_EXISTS = 1;
1724     $foo = $querySELECT_DB($XLAYOUT, $thisQueryFieldName, $thisQueryKey, $LDEBUG_FTN);
1725     if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; return $EXIT; }
1726 }
1727
1728 $thisQuery = "UPDATE $mailingTable SET $fieldName = '$fieldValue', $fieldName1 = '$fieldValue1' WHERE $thisQueryFieldName = '$thisQueryKey'";
1729
1730
1731 $RS = $Conn->Execute($thisQuery);
1732
1733 #if($foo = $RS;
1734 #foreach(keys($foo)) { print "----key = $ _ value = $foo{$ _} \n"; }
1735 #endif
1736
1737 if($RS) {
1738     $errors = $Conn->Errors();
1739     print $errors{"_"};
1740     foreach $err ($keys $errors) {
1741         print $err{"_"};
1742     }
1743     $sevenType = 'WRN_FUP'; $sevenReqResUP = (($sevenReqResUP == /badmob/) ? '' : /badmob/); $sevenDesc = "\<" . $thisRecordID . "\> (SLAYO
UT('thismailingpsn')) queryUPDATE_DB :: Cannot create RS: \($thisQuery\); $logEvent(\\"FH_log, $eventType, $sevenType, $sevenDesc, $user, $LDEBUG_FTN);
1744     $EXIT = 0;
1745     return $EXIT;
1746 }
1747
1748 $sevenType = 'INFO'; $sevenDesc = "\<" . $thisRecordID . "\> (SLAYOUT('thismailingpsn')) queryUPDATE_DB :: UPDATE SUCCESSFUL: \($thisQuery\); $log
Event(\\"FH_log, $eventType, $sevenType, $sevenDesc, $user, $LDEBUG_FTN);
1749
1750 return $EXIT;
1751 }
1752
1753 -----
1754
1755 sub querySELECT_DB (
1756     my($LAYOUT, $thisQueryFieldName, $thisQueryKey, $LDEBUG_FTN) = @_;
1757     my($DSN) = $LAYOUT{'thismailingpsn'};
1758     my $RS = undef;

```

```

1767 my $Conn = undef;
1768 my $Errors = undef;
1769 my $fieldNames = undef;
1770 my $PID = -1;
1771 my $recCount = 0;
1772
1773 # construct/open DSN
1774
1775 my $Conn = undef;
1776
1777 $Conn = new Win32::OLE("ADODB.Connection");
1778 $Conn->open($slayingDSN, "", "");
1779
1780 my($obj) = undef;
1781 $obj{"_exit_"} = 1;
1782
1783 # load data into recordset
1784
1785 my $mailingTable = $slaying{"thismailingtable"};
1786
1787 $thisQuery = "SELECT * FROM $mailingTable WHERE $thisQueryFieldName = '$thisQueryKey'";
1788
1789 $rs = $Conn->Execute($thisQuery);
1790
1791 if(!$rs) {
1792     $Errors = $Conn->errors();
1793     print "Errors:\n";
1794     foreach $error (keys %$Errors) {
1795         print $error->{description}, "\n";
1796     }
1797     $eventType = "FAIL"; $eventDesc = "\<"; $thisRecordID = "\> ($slaying{"thismailingDSN"}) querySELECT_DB :: Cannot create RS: \($thisQuery)\"; $logEvent("\FH_log, $eventType, $eventDesc, $user, $logDebug_FTN);
1798 }
1799
1800 my $i = undef;
1801
1802 for($i=0; $i < $rs->fields->count; $i++) { $fieldNames[$i] = $rs->fields->item($i)->name; }
1803
1804 my $RESULT_FLAG = 1;
1805
1806 if($rs->EOF) { $RESULT_FLAG = 0; }
1807
1808 $rs -> MoveNext;
1809
1810 if(!$rs->EOF) { $RESULT_FLAG = -1; }
1811 $rs -> MovePrevious;
1812
1813 if($RESULT_FLAG < 1) {
1814     (($eventReq'presup == "/badlog/?"'':'.badlog'); $FUPcount++; $eventType = "ERR_FUP"; $eventDesc = "\<"; $thisRecordID = "\> ($slaying{"thismailingDSN"}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG: \($thisQuery)"; $logEvent("\FH_log, $eventType, $eventDesc, $user, $logDebug_FTN);
1815 }
1816
1817 $obj{"_exit_"} = $RESULT_FLAG;
1818
1819 return $obj;
1820 }
1821
1822 while ( ! $rs->EOF ) {
1823     for($i=0; $i < $rs ->fields->count; $i++) {
1824         $obj{$fieldNames[$i]} = uc($rs->fields->item($i)->value);
1825     }
1826     $rs->MoveNext; $recCount++;
1827 }
1828
1829 $Conn->Close;
1830
1831
1832
1833

```

```

1834 return %DB;
1835 }
1836 }
1837 #
1838 # -----
1839 #
1840
1841 sub read_DSNTODAH (
1842
1843 my($DSN,$LGDBUG_FTN) = @_;
1844
1845 #print "DSN = $DSN \n";
1846
1847 $RS = undef;
1848 $Conn = undef;
1849 $Errors = undef;
1850 $fieldNames = undef;
1851 $PID = -1;
1852 $recCount = 0;
1853
1854 # construct/open DSN
1855
1856 $Conn = new Win32::OLE("ADODB.Connection");
1857 $Conn->Open($DSN);
1858 my($PFILE) = undef;
1859
1860 # Load data into recordset
1861
1862 $thisQuery = "SELECT * FROM Email";
1863
1864 $RS = $Conn->Execute($thisQuery);
1865
1866 if(!$RS) {
1867     $Errors = $Conn->Errors();
1868     print "Errors:\n";
1869     foreach $error (keys %$Errors) {
1870         print $error->Description, "\n";
1871     }
1872     $eventType = 'FAIL'; $eventDesc = "read_DSNTODAH: cannot create RS: \"$thisQuery\""; $logEvent("\r\n_log_7", $eventType, $eventDesc, $user, $LGDBUG_FTN);
1873 }
1874
1875 my $i = undef;
1876
1877 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1878 while ( !$RS->EOF ) {
1879     for($i=0; $i < $RS->Fields->Count; $i++) {
1880         $FILE[$recCount][$fieldNames[$i]] = $RS->Fields->Item($i)->Value;
1881     }
1882     $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
1883 }
1884
1885 $Conn->Close;
1886
1887 return $FILE;
1888 }
1889
1890 sub removeTabs {
1891     my($tabs) = @_;
1892     my($i) = 0;
1893     while ($i < $tabs->Count) {
1894         $tabs->Item($i) =~ s/\t//g;
1895     }
1896     return $tabs;
1897 }
1898
1899 # -----
1900
1901

```

P:\Development\PRAPPJ\hbp_extractOrder-WORKING\procOrder.pl
 Printed at 18:48 on 06 Feb 1998

```

1902 Stabs = s/\t//g;
1903
1904 return Stabless
1905
1906 }
1907
1908 }
1909
1910 }
1911
1912 sub subjectsynonymlookup (
1913   my($subject,$ptrHASHLAYOUT) = @_;
1914   my($LAYOUT) = $ptrHASHLAYOUT;
1915   my($mailingID) = -1;
1916   my($subjectmailingIDToken) = $1;
1917   my($subjectmailingIDToken) = $1;
1918   my($subjectmailingIDToken) = $1;
1919   my($subjectmailingIDToken) = $1;
1920   my($subjectmailingIDToken) = $1;
1921   my($subjectmailingIDToken) = $1;
1922   my($subjectmailingIDToken) = $1;
1923   my($subjectmailingIDToken) = $1;
1924   my($subjectmailingIDToken) = $1;
1925   my($subjectmailingIDToken) = $1;
1926   my($subjectmailingIDToken) = $1;
1927   my($subjectmailingIDToken) = $1;
1928   my($subjectmailingIDToken) = $1;
1929   my($subjectmailingIDToken) = $1;
1930   my($subjectmailingIDToken) = $1;
1931   my($subjectmailingIDToken) = $1;
1932   my($subjectmailingIDToken) = $1;
1933   my($subjectmailingIDToken) = $1;
1934   my($subjectmailingIDToken) = $1;
1935   my($subjectmailingIDToken) = $1;
1936   my($subjectmailingIDToken) = $1;
1937   my($subjectmailingIDToken) = $1;
1938   my($subjectmailingIDToken) = $1;
1939   my($subjectmailingIDToken) = $1;
1940   my($subjectmailingIDToken) = $1;
1941   my($subjectmailingIDToken) = $1;
1942   my($subjectmailingIDToken) = $1;
1943   my($subjectmailingIDToken) = $1;
1944   my($subjectmailingIDToken) = $1;
1945   my($subjectmailingIDToken) = $1;
1946   my($subjectmailingIDToken) = $1;
1947   my($subjectmailingIDToken) = $1;
1948   my($subjectmailingIDToken) = $1;
1949   my($subjectmailingIDToken) = $1;
1950   my($subjectmailingIDToken) = $1;
1951   my($subjectmailingIDToken) = $1;
1952   my($subjectmailingIDToken) = $1;
1953   my($subjectmailingIDToken) = $1;
1954   my($subjectmailingIDToken) = $1;
1955   my($subjectmailingIDToken) = $1;
1956   my($subjectmailingIDToken) = $1;
1957   my($subjectmailingIDToken) = $1;
1958   my($subjectmailingIDToken) = $1;
1959   my($subjectmailingIDToken) = $1;
1960   my($subjectmailingIDToken) = $1;
1961   my($subjectmailingIDToken) = $1;
1962   my($subjectmailingIDToken) = $1;
1963   my($subjectmailingIDToken) = $1;
1964   my($subjectmailingIDToken) = $1;
1965   my($subjectmailingIDToken) = $1;
1966   my($subjectmailingIDToken) = $1;
1967   my($subjectmailingIDToken) = $1;
1968   my($subjectmailingIDToken) = $1;
1969   my($subjectmailingIDToken) = $1;
1970   my($subjectmailingIDToken) = $1;

```

Page 31

For Developmental Psychology, please contact Order Manager at 1-800-354-9712.
Printed at 18:48 on 06 Feb 1999

```

1971 if ($f1empty) { chop($header); print $FW $header "\n"; }
1972 if ($AND_CONDITION ne "AND_RECORD_TYPE ne 'BODY'") { $record = &canonicalize($record); }
1973 print $FW $record "\n";
1974 }
1975 flock($FW, LOCK_UN);
1976 }
1977 }
1978
1979 -----END-----
1980

```


Claims

- 1 1. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction, and
4 based on the derived information, and
5 automatically generating commercial transaction data
6 in a format that is usable to automatically complete the
7 commercial transaction.
- 1 2. The method of claim 1 in which the commercial
2 transaction comprises an order for a product or service.
- 1 3. The method of claim 1 in which the e-mail
2 message comprises at least part of an e-mail sent to a
3 customer and responses of the customer to the e-mail.
- 1 4. The method of claim 1 in which the automatic
2 completion of the commercial transaction comprises order
3 fulfillment.
- 1 5. A machine-based method comprising
2 sending an e-mail message to a customer offering a
3 product or service for sale, the e-mail message comprising
4 locations for response by the customer indicating his
5 intention to order the product or service,
6 receiving from the customer an e-mail message that
7 includes the response,
8 based on the received e-mail, automatically
9 generating order information in a format usable
10 automatically by an order fulfillment system to cause the
11 order to be filled.
- 1 6. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction,
4 automatically identifying response information which
5 requires resolution of an issue with the source of the e-
6 mail message, and

7 automatically managing an e-mail dialog with the
8 source to resolve the issue.

1 7. The method of claim 6 in which at least some of
2 the e-mail dialog is performed automatically.

1 8. Software guided interactive e-mail dialogs to
2 resolve, on behalf of a vendor, customer issues that occur
3 in direct response e-mails that are automatically identified
4 as requiring a dialog.

1 9. A machine-based method comprising
2 automatically sorting e-mail messages, based on
3 response information contained in the messages, into e-mail
4 messages that can be processed automatically to generate
5 commercial transactions, e-mail messages in which the
6 response information is inadequate to permit generation of
7 commercial transactions, and e-mail messages that may be
8 subjected to exception handling to yield information that is
9 sufficient to generate commercial transactions.

1 10. A machine-based method comprising
2 analyzing an e-mail message to derive response
3 information concerning a commercial transaction, and
4 automatically generating a confirmatory e-mail
5 message to the source of the e-mail message confirming that
6 the commercial transaction has been or will be completed.

1 11. A machine-based method comprising
2 receiving inbound e-mail messages that result from
3 corresponding outbound e-mail messages associated with a
4 marketing program, the inbound messages containing response
5 information, each of the outbound messages being associated
6 with a distinct piece of the marketing program, and
7 automatically associating the response information
8 in each of the inbound messages with the corresponding
9 distinct piece of the marketing program.

1 12. The method of claim 11 in which the piece
2 comprises a marketing campaign or a marketing flight.

1 13. The method of claim 11 in which the inbound
2 messages contain information that links them to the
3 corresponding outbound messages, and the associating step
4 uses the link information.

1 14. The method of claim 13 further comprising
2 automatically parsing the inbound messages for order
3 information.

1 15. A machine-based method comprising
2 sending outbound e-mail messages associated with
3 commercial transactions,

4 storing information related to each of the outbound
5 messages in a database, the information being useful for
6 completing the commercial transactions, the information not
7 being contained in the outbound messages,

8 analyzing inbound e-mail messages that result from
9 the outbound messages and that contain response information
10 useful in completing the commercial transactions, and

11 automatically merging the response information with
12 corresponding information in the database for use in
13 completing the transactions.

1 16. A machine-based method comprising
2 sending outbound e-mail messages associated with
3 commercial transactions,

4 storing information related to each of the outbound
5 messages in a database, the information being useful for
6 completing the commercial transactions, the information not
7 being contained in the outbound messages,

8 analyzing inbound e-mail messages that result from
9 the outbound messages and that contain response information
10 useful in completing the commercial transactions,

- 11 identifying inbound e-mail messages that cannot be
- 12 processed automatically to generate the commercial
- 13 transactions, and
- 14 using the database information to assist in
- 15 exception handling of the identified inbound messages.

1/6

WILLIAM HERP

10

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
 SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
 TO: WILLIAM HERP
 SUBJECT: = NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
 BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:

- + PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
- + IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
- + HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
- + CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

=====

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

- A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
- B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
- C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
- D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
- E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR \$89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
 MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 3202.

=====

FIG. 1A

2/6

A: ** HARVARD BUSINESS REVIEW ON CHANGE **

PROVIDES LANDMARK IDEAS TO HELP YOU UNDERSTAND THE BEST WAYS FOR YOUR ORGANIZATION TO MANAGE CHANGE. INCLUDES ARTICLES BY JOHN KOTTER AND MORE. (240 PP/#8842/\$19.95)

B. ** HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT **

HIGHLIGHTS THE LEADING-EDGE THINKING AND PRACTICAL APPLICATIONS ON HOW COMPANIES GENERATE, COMMUNICATE, AND LEVERAGE KNOWLEDGE ASSETS. INCLUDES ARTICLES BY PETER DRUCKER, JOHN SEELY BROWN, AND MORE.
(240 PP/#8818/\$19.95)

C: ** HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH **

PRESENTS THE LATEST TACTICS FOR HELPING MANAGERS FIND AND EXPLOIT THE BEST OPPORTUNITIES FOR GROWTH AND PROFITABILITY. INCLUDES ARTICLES BY ARIE DE GEUS, JEFFREY RAYPORT, AND MORE.
(240 PP/#8850/\$19.95)

D: ** HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE **

OFFERS INSIGHT ON WHAT YOU NEED TO MEASURE AND HOW PERFORMANCE MEASURES CAN ALIGN AN ORGANIZATION AND BOOST PRODUCTIVITY. INCLUDES ARTICLES BY PETER DRUCKER, ROBERT KAPLAN AND DAVID NORTON, AND MORE.
(240 PP/#8826/\$19.95)

E: ** HARVARD BUSINESS REVIEW ON LEADERSHIP **

PRESENTS PROVEN FUNDAMENTALS OF LEADERSHIP AND CHALLENGES MANY LONG-HELD ASSUMPTIONS ABOUT THE TRUE SOURCES OF POWER AND AUTHORITY. INCLUDES ARTICLES BY JOHN KOTTER, JOSEPH BADARACCO, JR., AND MORE.
(240 PP/#8834/\$19.95)

F: PURCHASE THE EXECUTIVE COLLECTION (INCLUDES ALL FIVE TITLES) FOR JUST \$89- A SAVINGS OF MORE THAN 10%. (PRODUCT #7419BN)

[illegible]

FIG. 1B

3/6

[illegible]

◇ EACH PAPERBACK IS \$19.95 PLUS SHIPPING AND HANDLING ◇

IMPORTANT NOTE FOR OUR CUSTOMERS OUTSIDE THE U.S.: PURCHASERS ARE RESPONSIBLE FOR ALL DUTIES, TAXES, BROKERAGE FEES, AND/OR IMPORT FEES IMPOSED BY THE COUNTRY OF IMPORT. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO YOUR ORDER. DELIVERY TO CANADA: \$14.00 FOR THE FIRST TITLE, \$2.00 FOR EACH ADDITIONAL TITLE. INTERNATIONAL DELIVERY OUTSIDE NORTH AMERICA: \$20.00 FOR THE FIRST TITLE, \$5.00 FOR EACH ADDITIONAL TITLE.

PLEASE ALSO REVIEW AND UPDATE THE ADDRESS INFORMATION BELOW SO THAT WE CAN PROCESS YOUR REQUEST PROMPTLY.

FIRST NAME: [WILLIAM]
LAST NAME: [HERP]
TITLE: [PRESIDENT]
COMPANY: [E-CARE GROUP INC.]
DEPARTMENT: []
ADDRESS1: [1646 MASSACHUSETTS AVE.]
ADDRESS2: []
ADDRESS3: []
CITY: [LEXINGTON]
PROVINCE/STATE: [MA]
POSTAL/ZIP CODE: [2173]
COUNTRY: []
PHONE: []
FAX: []
EMAIL: [WHERP@E-CARE.COM]

[[891270||3202]]

FIG. 1C

4/6

WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: MONDAY, FEBRUARY 08, 1999 8:25 PM
TO: WILLIAM HERP
SUBJECT: ** A FREE NO-OBLIGATION TRIAL FROM HARVARD BUSINESS REVIEW

FROM THE DESK OF LAURA WINIG
HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS

MONDAY, FEBRUARY 8, 1999

~INTRODUCING~

BENCHMARKING A NEW THREE-PART VIDEO SERIES FROM HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION

TO TAKE *BENCHMARKING* FOR A NO-OBLIGATION 14-DAY TEST DRIVE, SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE

DEAR WILLIAM HERP:

INTERESTED IN UNEARTHING NEW IDEAS AND UNCONVENTIONAL SOLUTIONS FOR THE CHALLENGES FACING YOUR COMPANY? HERE AT THE PUBLISHING ARM OF HARVARD BUSINESS SCHOOL, WE'VE CREATED AN EXCITING NEW PROGRAM THAT CAN SHOW YOU HOW SOME LEADING COMPANIES USE BENCHMARKING -- STUDYING AND EMULATING TOP PERFORMERS INSIDE, AND OUTSIDE, THEIR INDUSTRIES -- TO ELIMINATE LONG-STANDING PROBLEMS AND BECOME TOP PERFORMERS.

DISCOVER HOW NEW PRACTICES CAN BE APPLIED TO YOUR ORGANIZATION -- WITH IMPRESSIVE AND MEASURABLE RESULTS -- IN BENCHMARKING, AN INNOVATIVE THREE-PART VIDEO SERIES. WE'LL TAKE YOU DEEP INSIDE PROFILED COMPANIES SUCH AS MOBIL OIL, GTE, AND SUNHEALTH TO LEARN HOW THEY IDENTIFIED "BEST OF CLASS" COMPANIES TO BENCHMARK IN ORDER TO IMPROVE THEIR OWN PERFORMANCE.

YOU'LL SEE HOW BENCHMARKING CAN GIVE YOUR TEAM A COMMON RALLYING POINT AND MOTIVATE COORDINATED ACTION. YOU'LL LEARN HOW TO IDENTIFY PROCESSES TO BENCHMARK, HOW TO FIND THE RIGHT PARTNER, AND HOW TO INITIATE THE FIRST STEPS (EVEN ON A LIMITED BUDGET). YOU'LL FIND OUT HOW TO IDENTIFY NOVEL OPPORTUNITIES, HOW TO STRUCTURE YOUR EFFORTS FOR SUCCESS, EVEN PROPER BENCHMARKING ETIQUETTE. EACH CONCEPT IS CLEARLY EXPLAINED AND ILLUSTRATED TO FACILITATE IMPLEMENTATION.

BENCHMARKING FOR CONTINUOUS IMPROVEMENT, BENCHMARKING CORE PROCESSES, AND BENCHMARKING OUTSIDE THE BOX BRING YOU FIRSTHAND COMMENTARY FROM SENIOR EXECUTIVES, INDUSTRY EXPERTS, AND FRONT-LINE PERSONNEL IN A FAST-PACED DOCUMENTARY STYLE THAT GENERATES INTEREST, UNDERSTANDING, AND ENTHUSIASM FOR THESE IMPORTANT IDEAS. THESE VIDEOS WILL STIMULATE DISCUSSION AND PROVIDE GUIDELINES TO HELP YOU DEVELOP AN ACTION PLAN FOR YOUR ORGANIZATION.

MAY I SEND YOU BENCHMARKING FOR A FREE, NO-OBLIGATION TRIAL? SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE AND WE'LL SEND YOU THE PROGRAM TO TRY WITH OUR COMPLIMENTS. WE'LL SEND YOU THIS INNOVATIVE SERIES RIGHT AWAY. AFTER 14 DAYS, WE WILL MAIL YOU AN INVOICE FOR \$1190 (A SAVINGS OF \$595 VERSUS THE INDIVIDUAL VIDEO PRICE OF \$595 EACH).

IF YOU ARE NOT COMPLETELY SATISFIED WITH BENCHMARKING SIMPLY RETURN IT TO US. YOU WILL OWE NOTHING. WHY WAIT TO LEARN HOW SUCCESSFUL CHANGE MANAGEMENT CAN DRAMATICALLY ENHANCE YOUR ORGANIZATION'S PERFORMANCE?

SINCERELY,
LAURA WINIG

FIG. 2A

5/6

DIRECTOR

P.S. IF YOU PREFER, PRINT OUT THIS INVITATION, INITIAL IT AT THE TOP, (PLEASE VERIFY YOUR SHIPPING ADDRESS IS CORRECT AS LISTED ABOVE -- WE MUST HAVE A STREET ADDRESS FOR SHIPMENT) AND FAX IT TO 617-496-1029, OR SIMPLY CALL 1-800-668-6780. PLEASE BE SURE TO MENTION PRIORITY CODE 3275.

CONTACT INFORMATION

BELOW PLEASE FIND THE CONTACT INFORMATION WE CURRENTLY HAVE ON FILE. IF THIS INFORMATION IS NOT CORRECT, PLEASE MAKE YOUR EDITS BETWEEN THE APPROPRIATE BRACKETS AND RETURN - VERBATIM - AS PART OF YOUR REPLY E-MAIL. PLEASE INDICATE ANY ADDRESS CHANGE BY INCLUDING THE WORDS "ADDRESS CHANGE" AT THE TOP OF YOUR ORDER-REPLY. IF YOU WISH TO UNSUBSCRIBE FROM SPECIAL OFFER MAILINGS, PLEASE REPLY TO THIS E-MAIL MESSAGE WITH THE WORD "UNSUB" AT THE TOP OF YOUR REPLY.

BILLING ADDRESS

BILL FIRST NAME:	[WILLIAM]
BILL LAST NAME:	[HERP]
BILL TITLE:	[PRESIDENT]
BILL COMPANY:	[E-CARE GROUP INC.]
BILL DEPARTMENT:]	
BILL ADDRESS1:	[1646 MASSACHUSETTS AVE.]
BILL ADDRESS2:]
BILL ADDRESS3:]
BILL CITY:	[LEXINGTON]
BILL PROVINCE/STATE:	[MA]
BILL POSTAL/ZIP CODE:	[02173]
BILL COUNTRY:]
BILL PHONE:]
BILL FAX:]
BILL EMAIL:	[WHERP@E-CARE.COM]

SHIPPING ADDRESS (IF DIFFERENT)

SHIP FIRST NAME:]
SHIP LAST NAME:]
SHIP TITLE:]
SHIP COMPANY:]
SHIP DEPARTMENT:]
SHIP ADDRESS1:]
SHIP ADDRESS2:]
SHIP ADDRESS3:]
SHIP CITY:]
SHIP PROVINCE/STATE:]
SHIP POSTAL/ZIP CODE:]
SHIP COUNTRY:]
SHIP PHONE:]
SHIP FAX:]
SHIP EMAIL:]

[[891270||3275]]

68 70 72

FIG. 2B

6/6

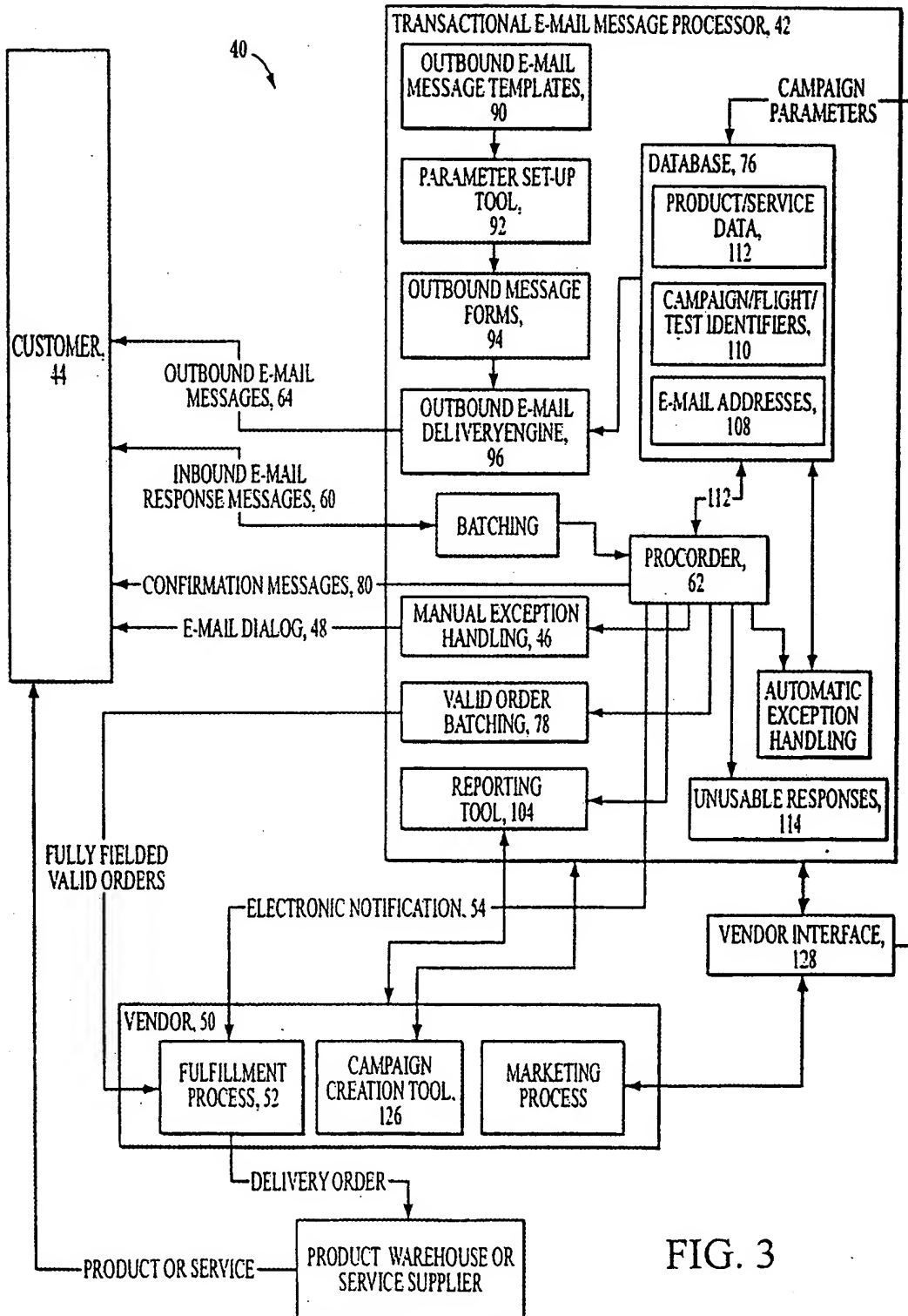


FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/19403

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/60

US CL : 705/1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/1,709/206,207,379/193,707/505

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DIALOG, WEST.EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,724,424 A (GIFFORD) 03 Mar 1998, Fig 1[68,64,66,67,62,200]; col 4 lines 48-53; col 5 lines 22-29; Fig.3[5,6,7]; col 5 lines 18-19;col 5 lines 34-37;Fig 5[18]; Fig 4[15]; col 5 lines 49-57; Fig 6[19-25];col 5 lines 29-44;Fig 4[8-17];Fig 2[1-3];Fig 3[3];col 5 line 25-col 6 line 5];Fig. 5;col 6 lines 20-29;col 6 lines 39-42;Fig.3[3]	1-16



Further documents are listed in the continuation of Box C.



See patent family annex.

*

Special categories of cited documents:

A

document defining the general state of the art which is not considered to be of particular relevance

E

earlier document published on or after the international filing date

L

document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

O

document referring to an oral disclosure, use, exhibition or other means

P

document published prior to the international filing date but later than the priority date claimed

T

later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X

document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y

document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

Z

document member of the same patent family

Date of the actual completion of the international search

19 AUGUST 2000

Date of mailing of the international search report

02 OCT 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Tod Swann

Telephone No.

Regina Logan

